

AUTOMATED FINGERPRINT ACTIVATED DOOR LOCK

by

**Jean Eric V. Agena
Karl Lester A. Co
Jon Remon D. Loon
Eliseo G. Noble Jr.
Judy Ann U. Rodriguez**

A Design Documentation Submitted to the
School of EE-ECE-CoE
in Partial Fulfillment of the Requirements for the Program
Bachelor of Science in Computer Engineering

Mapúa Institute of Technology

April 2008

APPROVAL SHEET

This is to certify that this study entitled “**Automated Finger Print Activated Door Lock**” prepared by **Jean Eric V. Agena, Karl Lester A. Co, Jon Remon D. Loon, Eliseo G. Noble Jr., Judy Ann U. Rodriguez** in partial fulfillment of the requirements for the degree **Bachelor of Science in Computer Engineering** have been supervised the preparation of and read the design documentation and hereby recommended for final examination by the Oral Examination Committee.

Ms. Angelina R. Hernandez
Non Technical Reader

Engr. Danilo R. Tiongco
Design Adviser

As members of the Oral Examination Committee, we hereby **APPROVED** this design study which was presented before a Panel of Examiners of the School of EE-ECE-CoE on **April 1, 2008**.

Engr. Jojo T. Sy
Panel Member 1

Engr. Joyce M. Santos
Panel Member 2

Engr. Gino Paolo Luis R. Villanueva
Panel Member 3

Accepted in partial fulfillment of the requirements for the degree **Bachelor of Science in Computer Engineering**.

Dr. Felicito S. Caluyo
Dean, School of EE-ECE-CoE

ACKNOWLEDGEMENT

This project will not come in reality without the help and support of many people. The following deserved to be acknowledged for without them, this accomplishment would be nothing.

To Almighty God for without His absolute power and for giving us sufficient intelligence, to conceptualize and create this project. We thank Him for bestowing us his blessings is advocating us to complete this course.

To our parents and family, for being supportive and understanding while we are doing this project.

To Engr. Noel Linsangan for guiding us in our course.

To our Design Adviser, Engr. Danilo S. Tiongco, for giving us knowledge in dealing with the concepts of our project for sharing his expertise and ideas from the start until the completion of this design.

To the professional people who guided us in relation to our project.

To Engr. Jojo T. Sy and Engr. Giovanni D. Fernandez, for clearing/initiating some vital points in our project design.

Lastly, to our friends, especially Ms. Kathleen May Cristine Sagun, for sharing her knowledge and inspiring us to create and finish this project.

TABLE OF CONTENTS

TITLE PAGE	i
APPROVAL SHEET	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	Viii
Chapter 1: INTRODUCTION AND REVIEW OF RELATED LITERATURE AND STUDIES	
Research Setting	1
Review of Related Literature and Related Studies	2
Conceptual Framework	8
Statement of the Problem	9
Objective of the Study	9
Significance of the Study	10
Scope and Delimitation	11
Definition of Terms	12
Chapter 2: METHODS AND PROCEDURES	
Research Design	15
Design Procedure for Actual Design	18
Hardware Design	18
Schematic Diagram	19
Hardware Components	20
Software Design	25
System Flowchart	26
Chapter 3: PRESENTATION AND INTERPRETATION OF DATA	
Accuracy of the Design	31

Chapter 4: CONCLUSION AND RECOMMENDATION

Conclusion	36
Recommendation	37
BIBLIOGRAPHY	38
APPENDICES	39
APPENDIX A – List of Materials	39
APPENDIX B – Pictures of the Design	41
APPENDIX C – Source Code	43
APPENDIX D – MAX 232 Datasheet	72
APPENDIX E – Z86E08 Datasheet	90
APPENDIX F – User Manual	104

LIST OF TABLES

Table 1: Comparison of Biometric Devices	4
Table 2: First Trial Table	31
Table 3: Second Trial Table	32
Table 4: Third Trial Table	33
Table 5: Fourth Trial Table	34
Table 6: Biometric Device Trial	35

LIST OF FIGURES

Figure 1: Conceptual Framework	8
Figure 2: Data Gathering Procedure Flowchart	16
Figure 3: System Flow Diagram	17
Figure 4: System Hardware Block Diagram	18
Figure 5: Door Lock / Unlock Triggering Flowchart	19
Figure 6: Hardware System Flowchart	24
Figure 7: Main menu Activate System Flowchart	26
Figure 8: Log-in Flowchart	27
Figure 9: Administration Control Flowchart	28
Figure 10: Database Entity-Relationship Diagram	29

ABSTRACT

The automation of fingerprint activated door lock aims to interface a bio-metric reader, specifically a fingerprint scanner, and the creation of an application program that will enable automatic locking and opening of a door in a specific room. This device will replace the use of keys, passwords and cards. It can be implemented in private offices and schools. The design is composed of three vital parts, the biometric reader (fingerprint scanner), the software installed in a computer that will enable the locking and unlocking of the door, and the circuit interfaced to the computer to trigger the locking and unlocking of the door. Visual Basic.Net is used to create the software while the circuit used Z86E08 microcontroller unit to handle the address received from the computer to trigger relays for locking and unlocking of the door. Through the use of this system, securing the access to establishments while providing convenience and efficiency in entering a room can be achieved.

Keywords: Biometric, Fingerprint, Software, Microcontroller unit, Relay, Z86E08

Chapter 1

INTRODUCTION AND REVIEW OF RELATED LITERATURE AND STUDIES

Research Setting

Technologies nowadays are advanced and so many countries have switched from manual system to electronic machines. Different devices are invented for the betterment of humanity and one of its benefits is in security. And when it comes to security, most progressive countries use bio-metric locks, an alternative to manual and card locks to secure homes and offices, as well as business establishments. This has lessened minor inefficiency in most firms. Such inefficiency even in its smallest measures can affect the productivity of each individual. One of the most frustrating things to realize is when an individual/person has forgotten his keys, locked himself out of his house, had them lost or stolen, or that an intruder has broken in. Another example of this is when a building has many rooms and uses the old fashioned door locks. Imagine the number of keys that one has to keep and remember. One or two doors will take some time to open especially when those keys are not labeled. One more scenario is when a room is restricted and only authorized person are allowed to get in, it is a hassle to monitor such rooms considering that the room should be opened for easy access to those persons. In some establishments like schools, they have routine access to rooms that should be locked and opened at a certain hour or at the next hour. This is done mostly by janitors, security guards or employees assigned in the said area which oftentimes they forget to do because they have other responsibilities in the said building. This scenario is evident at Mapua Institute of Technology which was the basis of the researchers' design. The researchers took the laboratory schedule as a case. During a laboratory class, the professor had to fetch an

employee just to open the door. It will take the employee about 10 to 15 minutes just to open a door. Likewise, after the class, it would take the same amount of time and effort just to close and lock the door and so on for the next class. This set-up happens everyday for all school terms. This establishment can implement automation of door locks using biometric sensors as an alternative to manual and card locks. Moreover, this will also provide efficiency through easy access to rooms especially for scheduled use of rooms. This will ensure authorized entry and, of course eliminate the need for keys.

Review of Related Literature and Related Studies

PIC-BASED DOORLOCK SYSTEM AND BIO-METRIC DEADBOLT LOCK

The PIC-based door lock system designed by A. Bitoon, et. al., (September 2003), stated that, “PIC-based door lock system provides a means of replacing old fashioned locks using keys by means of sensors and readers. With this door lock system homes and establishments can avail of better safety and security. It uses components such as a keypad for password input and Programmable Integrated Circuit microcontroller as to control the functions of the system.”

Another study was done by S. Viscusi (2006) which is the Sequiam Bio-metric Door Lock. The Bio-metric Deadbolt lock aside from being a stand alone lock replaces the use of keys to enter a room. This provides an authorized entry to prevent intruders to break in homes. A swipe of an authorized finger through a scanner grants access. It uniquely solve the problems of homeowners in securing their homes while having the easiest way to enter their houses.

Based on this study, the writers learned that a microcontroller can also be used as a device for locking and unlocking doors. This concept gave the researchers the idea to create a device that will enable automatic locking and unlocking of door that can be used in homes and establishments that can avail of better safety and security. Based on an article published in Blackheath, South Africa last 2006, Bio-metric lock provides a more secured access to homes, as well as the easiest way to enter a room or an establishment. From the previous study, the researchers came up with the idea of using fingerprint scanner instead of keypads. This will result to a more convenient way to enter a door. Static pins are no longer needed for a hassle free access to a room. Moreover, this article gave the researchers the idea that instead of using a stand-alone device; why not further utilize its purpose by interfacing it to a computer software. This can be the answer to the limited storage installed in the device. Having the scanner interfaced with the computer proposes more memory storage and capacity which is a good concept in continuing with the design. The researchers realized that a fingerprint scanner can be an effective means of securing the access on rooms while providing the most efficient and easiest way of entering an establishment.

The article entitled, “The State of the Bio-metric Industry: The Search for Security and Convenience”, written by Peter Burgess, Technical Manager of RSA Security in America (2001), discussed the prevalent industry of bio-metrics. The author said “bio-metrics apply to a broad range of electronic techniques that use unique physical characteristics of human beings as a means of authentication. Usually these are considered the domain of James Bond films or ultra-sensitive military installations. However, the current range of its application is now rapidly increasing that is why more

organizations whether from the government, school, or business districts need it for positive identification.”

The use of bio-metric devices falls into 2 main categories: law enforcement (government) and building access. Authentication purposes, convenience and password-replacements have been its strongest drivers.

One of the primary drivers for bio-metrics is its ability to provide a viable alternative to the ubiquitous password. Passwords are now widely recognized as an extremely weak form of authentication. In fact, up to 50% of costly help desk calls are from users who have forgotten or misplaced their passwords.

Authentication by biometric verification is becoming increasingly common in corporate and public security systems, consumer electronics and point of sale (POS) applications. In addition to security, the driving force behind biometric verification has been convenient.

Biometrics:	Universality	Uniqueness	Permanence	Collectability	Performance	Acceptability	Circumvention
Face	H	L	M	H	L	H	L
Fingerprint	M	H	H	M	H	M	H
Hand veins	M	M	M	M	M	M	H
Iris	H	H	H	M	H	L	H
Retinal scan	H	H	M	L	H	L	H
Signature	L	L	L	H	L	H	L
Voice	M	L	L	M	L	H	L

Table 1 – Comparison of Biometric Devices

Table 1 shows a comparison of existing bio-metric devices. A. K. Jain ranks each bio-metric device based on the categories as being low, medium or high. A low ranking

category indicates poor performance in the evaluation criterion, whereas a high ranking category indicates a very good performance. It is possible to understand if a human characteristic can be used for bio-metrics in terms of parameters: universality, uniqueness, permanence, collectability, performance, acceptability and circumvention.

1. **Universality:** each person should have the characteristic
2. **Uniqueness:** how well the bio-metric separates individually from another.
3. **Permanence:** measures how well a bio-metric resists aging.
4. **Collectability:** ease of acquisition for measurement.
5. **Performance:** accuracy, speed, and robustness of technology used.
6. **Acceptability:** degree of approval of a technology.
7. **Circumvention:** ease of use of a substitute.

Furthermore, a bio-metric device that uses fingerprints show more pleasing results among the rest. Other than the parameters stated in the table the researchers chose a fingerprint over other bio-metrics because of its cost and portability. A finger print scanner can be easily installed and integrated to a program based from its application. It is cost efficient and acceptable by the industry. While all types of biometrics are likely to grow as costs reduced, technology improves and demand increases, fingerprint scanning will continue to hold the largest market share and offers the best trade-off between cost and reliability/user-friendly.

The promise of bio-metrics combines both security and convenience because the user does not have to carry any additional device or remember a static pin. Examples from specific vertical markets have shown significant demand for bio-metrics todate that provide perfect illustrations. On financial services, it is faster and simpler for traders on a

hectic trading floor to log into the network with a fingerprint scanner than having to remember or enter a 6-digit password. For healthcare, the primary objective is to enable clinicians to quickly access electronic patient records in campus-type environments. The quicker they are in and out of the network, the sooner they can care for patients. Both of these examples also show how organization can justify an investment in bio-metric solutions if users can execute more transactions per day or visit with more patients per day.

Peter Burgess stated in 2001 that bio-metrics promise both security and convenience. The use of such device proposes a viable solution to an efficient means on building access while providing positive authentication. Implementing bio-metric system especially to technologically advanced establishments offers not only security and convenience but also efficiency in terms of time and productivity.

According to this article, fingerprint scanner is the most accepted biometric device that the researchers can start to work with. It offers a better trade-off between cost and reliability. Based from the studies/articles that the researchers have gathered, a system basically a finger scanner interfaced with a computer software to control a door (locking and unlocking) can be a good design which can be an innovation on old fashioned doors.

FINGERPRINT SCANNER

Bio-metrics is the science and technology of measuring and analyzing biological data. In information technology, bio-metrics refer to technologies that measure and analyze human body characteristics, such as fingerprints, eye retinas and irises, voice patterns, facial patterns and hand measurements for authentication purpose.

Bio-metric devices such as fingerprint scanners consist of:

- a reader or scanning device ;
- software that converts the scanned information into digital form and compares match points; and,
- a database that stores the biometric data for comparison.

To prevent identity theft, biometric data is usually encrypted when gathered. Here is how bio-metric verification works on the back end: to convert the biometric input, a software application is used to identify specific points of data as match points. The match points in the database are processed using an algorithm that translates that information into a numeric value. The database value is compared with the biometric input that the end user has entered into the scanner and authentication is either approved or denied.

Conceptual Framework

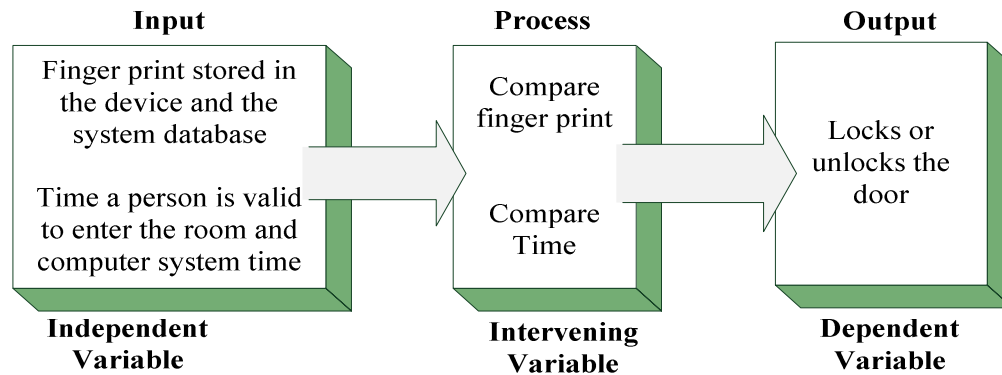


Figure 1 – Conceptual Framework

Figure 1 illustrates the Conceptual Framework that the researchers will work on. Two concepts will be considered in the design, one for the verification of authorized entry and another for the automation of the locking system. Since the design will focus on a routine based operation the system will take the time as an input. In view of the fact that the design will be computer-based, the computer's system time is an input for the project. Also, the schedule for laboratory classes will be taken as the basis for timing the device. Class intervals, to be specific, will be considered for comparison of time to automatically lock the door. For the verification of authorized entry, fingerprints are taken as an input to the system which will be compared to the stored fingerprints in the memory.

The said inputs will be evaluated by the system. The class interval and the system time will be compared and if it matches will result to automatic locking of the door. For the authorized entry, thumb prints will be compared. A match of prints will either open or close the door.

The output of the system will depend on the process made in relation with the input made by the user, as well as the system; the locking and unlocking of the door is then considered as the dependent variable of the system. This will fit to solve the problem if the functionality of the system is met.

Statement of the Problem

What can be implemented to innovate the scheduled manual locking of doors?
What means can be used to eliminate the use of passwords, keys and cards that maybe forgotten, lost or stolen which is secured for use in private establishments, like schools?
The specific problems that the design need to solve are as follows:

1. What device can be used to replace the use of keys, passwords and cards?
2. In what way can a user access a room with convenience and efficiency?
3. What can be implemented to provide a secured access to rooms with minimum time wasted?

Objective of the Study

The objective of this study is to interface a biometric reader and develop an application program that will enable the automatic locking and opening of a door in a specific time interval. This device should replace the use of keys, passwords and cards. This should provide a convenient, secured and authorized entrance to a room or establishment.

Significance of the Study

One of the most frustrating things to realize is when a person has forgotten his keys, locked himself out of his house or office, has them lost or stolen, or that an intruder has broken in. These worries most individuals who still use keys for their door locks. Some establishments use smart cards that are often lost or left inside a room. In some cases, there are employees in charge of keys who do the locking and unlocking of rooms on a routinary basis. Oftentimes, if not forgotten employees went out or do other things, which in return, can be the cause of delay and hassle to persons who will be using the room. These are frequent distractions in such scenarios. This design provides convenience in using a room without the need to remember any password, find keys or bring cards that are often misplaced. As an authorized user, there won't be a problem for delay in using a room. For additional feature, this will also help in monitoring the person who has access to the room.

By implementing the system, it also offers a way for users (students/instructors) to exercise discipline and to value time. Other influences that can affect how efficiently one use his time can be equivalent to how many distractions he has around him. The most efficient use of time involves minimizing distractions and discouraging procrastination. Through time restrictions and access monitoring, procrastination can be discouraged. Minimizing time wasted offers more time for learning. Students will focus and concentrate more on working with given tasks.

Scope and Delimitations

The bio-metric fingerprint scanner is interfaced with an application program that will enable an automatic locking and unlocking of doors on a certain time interval. The following are the scope of the system:

1. The system uses a Universal Serial Bus bio-metric fingerprint scanner.
2. The scanner is an angle sensitive device which may require users for multiple fingerprint reading to lock/unlock door.
3. The application software will determine if the prints are valid or not.
4. The program will also verify if the access to a room is valid for that day and time.
5. The system uses a serial port to communicate with the circuit that locks/unlocks the door.
6. The software is capable of monitoring room access, adding valid users and registering fingerprints.
7. The application software is capable of scheduling access validity date and time for each user based from a pre-set schedule hard coded to the software.
8. The system software is designed to work in windows 2000 sp4 to XP.
9. The schedule of rooms is patterned with the institute's schedule of classes.
10. One hour and a half is equivalent to five minutes in the system.
11. A grace period is included in the five-minute class period, which is one third of the time based on the school's schedule.
12. A scheduled access to the room is only valid during the grace period, meaning unlocking of the door can only be done during the grace period. After the grace period, the professor's authority to unlock the door expires.

The delimitations of the system are as follows:

1. Only through the use of a computer and the fingerprint scanner is capable of locking and unlocking of the door.
2. The circuit can only be interfaced to the system through a serial port or a USB to RS232 plug.
3. The circuit only receives data and is not capable of transmitting information to the computer.
4. The system is not capable of knowing the number of people entering or leaving the room.
5. The design will not function in case of power interruption.
6. External interruption cannot be controlled by the system. In effect, this might cause some malfunction or some parts of the design might not function properly.

Definition of Terms

1. Alternating Current (AC) - is an electrical current whose magnitude and direction vary cyclically with time, as opposed to direct current, whose direction remains constant. The usual waveform of an AC power circuit is a sine wave, as these results in the most efficient transmission of energy. <*Britannica Encyclopedia, 2003*>.
2. Ampere – The ampere is a unit of electric current, or amount of electric charge per second. <*Britannica Encyclopedia, 2003*>.
3. Biometric – is the science that involves analysis of biological characteristics. In computer industry, it means the verification of peoples' identities' using their unique characteristics like fingerprints. <*Microsoft Encarta Encyclopedia, 2003*>.

4. Bio-metric lock – uses eyes, face recognition, voice, fingerprints or vein checks to either lock or unlock doors. <Microsoft Encarta Encyclopedia, 2005>.
5. Central Processing Unit (CPU) – is a computational and control unit of a computer; the device that interprets and executes instructions. <Microsoft Encarta Encyclopedia, 2005>.
6. Crystal Oscillator – a device in which the frequency is controlled by a piezo-electric crystal. A crystal oscillator may require controlled temperature because its operating frequency is a function of temperature. <Microsoft Encarta Encyclopedia, 2003>.
7. Direct Current (DC or "continuous current") - is the constant flow of electric charge. <Microsoft Encarta Encyclopedia, 2003>.
8. Electric Current – is the flow of electric charge. <Microsoft Encarta Encyclopedia, 2003>.
9. Electric Power – is defined as the rate at which electrical energy is transferred by an electric circuit. <Britannica Encyclopedia, 2003>.
10. Electricity – is a general term for a variety of phenomena resulting from the presence and flow of electric charge. <Britannica Encyclopedia, 2003>.
11. RAM (Random Access Memory) - a computer memory on which data can be both read and written and on which the location of data does not affect the speed of its retrieval; especially RAM that acts as the main storage available to the user for programs and data. <Merriam – Webster Dictionary, 2006>.
12. ROM (Read Only Memory) - a usually small computer memory that contains special-purpose information (as a program) which cannot be altered. <Merriam – Webster Dictionary, 2006>.

13. SPDT – also known as Single Pole Double Throw consists of 1 common terminal, 1 normally closed terminal, and one normally opened terminal. <*Microsoft Encarta Encyclopedia, 2003*>.
14. Stepper motor – tool or shaft that has permanent magnets attached to it. Around the body of the motor is a series of coils that create magnetic field and interacts with the permanent magnets. When these coils are turned on and off, the magnetic field causes the rotor to move. <*Microsoft Encarta Encyclopedia, 2003*>.
15. Uninterruptible Power Supply (UPS) - provides correct working voltages to the circuits of an electronic device in spite of interruptions to the incoming electrical power supply from the grid. <*A Dictionary of Computing, 2004*>.
16. Universal Serial Bus (USB) - a standardized serial computer interface that allows simplified attachment of peripherals especially in a daisy chain. <*Merriam – Webster Dictionary, 2006*>.
17. Volt – the volt or V is the International System derived unit of electric potential difference or electromotive force. <*Britannica Encyclopedia, 2003*>.
18. Voltage – sometimes also called electric potential or electrical tension is the difference of electrical potential between two points of an electrical or electronic circuit expressed in volts. <*Britannica Encyclopedia, 2003*>.

Chapter 2

METHODS AND PROCEDURES

Research Design

The researchers use descriptive and experimental methodologies as an approach to solve the problems stated in the previous discussion. These two methods are important for the researchers to study factors that will or will not affect the implementation of the solution to problems. Descriptive research was used by the researchers to obtain information concerning the current status of the phenomena to describe "what exists" with respect to variables or conditions in a situation. By observation and other ways to determine, analyze and answer the problem, the researchers have come up with a viable solution. This solution is based on the review of the related literature and studies, as well as to the possible solutions cited by the researchers before starting the construction of the prototype. An attempt by the researchers to maintain control over factors that may affect the result of an experiment is the reason why the researchers also prefer to use the Experimental Research. In doing this, the researchers attempted to determine or predicted what may occur. An experimental design or construction of a prototype is important to determine the procedures that enable the researchers to test the solution by reaching valid conclusions about relationships between independent and dependent variables.

From the given problem, the possible solutions must be first analyzed. There will be many solutions to a particular problem. All the requirements and conditions of the system must be taken into account to identify the vital system components to be used and

for what purpose. To make the work efficient a procedure was followed by the researchers for an organized implementation of the solution to the problem.

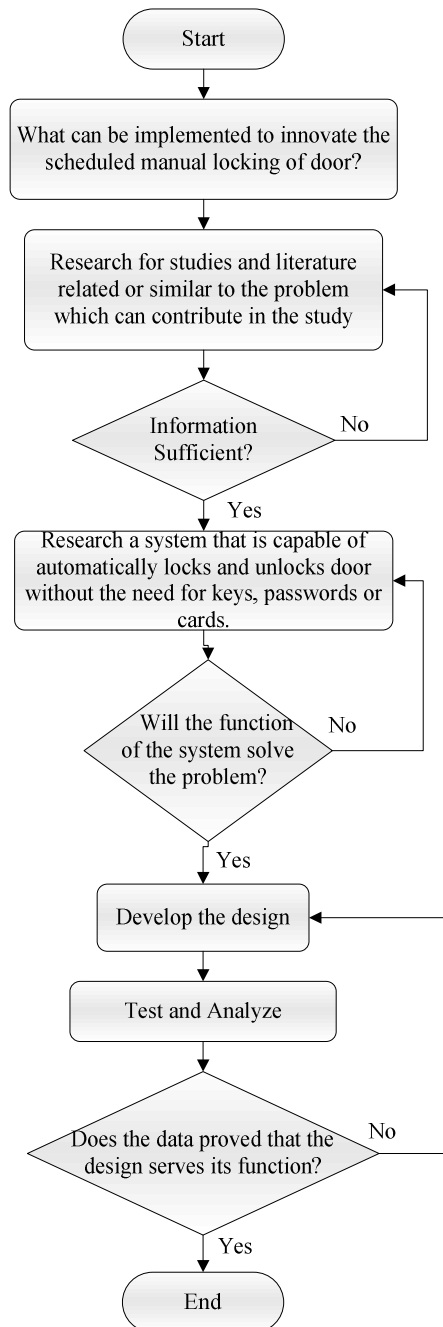


Figure 2 – Data Gathering Procedure Flowchart

To start the process, the researchers have to determine the problem that needs to be solved. After which, a research should be conducted to find possible ways on solving the problem. Literatures and other studies can be reviewed to contribute in choosing the most viable solution to the problem. If information gathered is sufficient, a research should be conducted again to further analyze the chosen solution to the problem. If the system functions are met the group is likely to proceed in developing the system.

Testing and analyzing the system after it was developed is vital in the process of creating the system. It will help distinguish what needs to be removed and revised in the system to meet the objective of the design based on Figure 2.

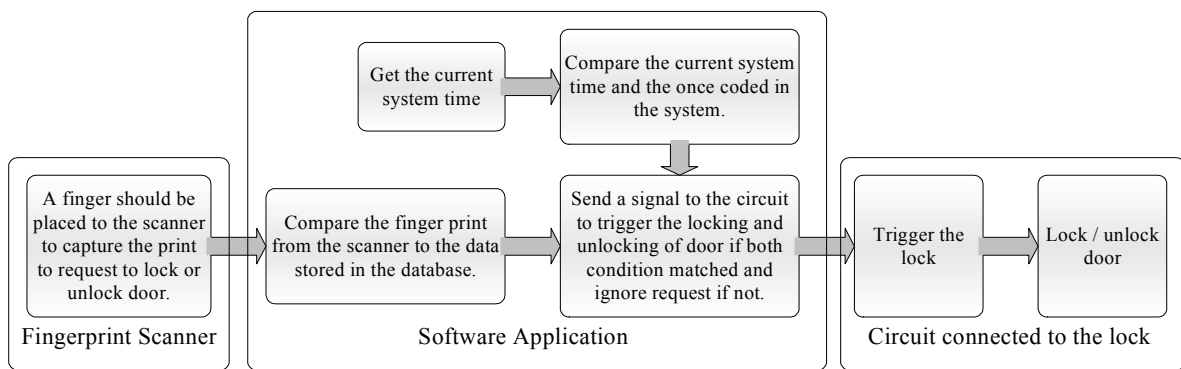


Figure 3 – System Flow Diagram

Figure 3 shows the flow of the system. Based from the diagram, the application software is responsible for processing input values. If input values matched the pre-set conditions and information contained in the database, it will then send an address to the circuit to trigger the locking and unlocking of door. In any case if those conditions are not met the software will ignore the request.

Accuracy testing will be used to test the functionality of the system. If the system is able to lock or unlock the door then this solution will prove to fit in answering or solving the problem.

Design Procedure for Actual Design

HARDWARE DESIGN

Block Diagram

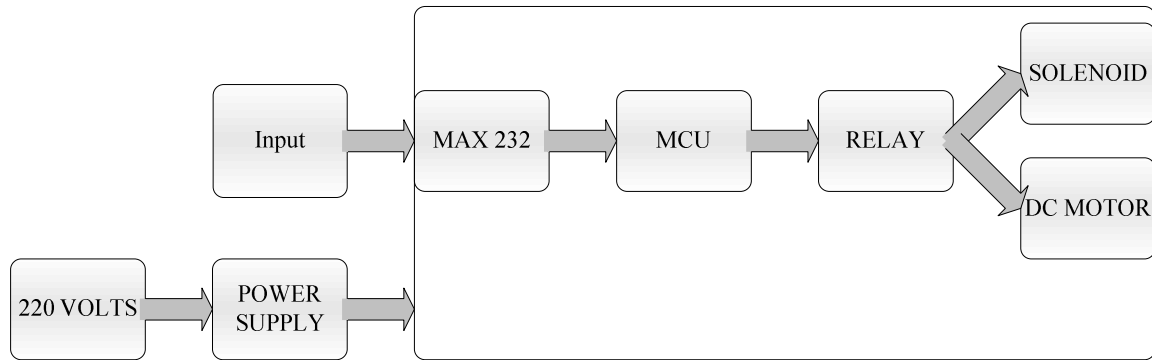


Figure 4 – System Hardware Block Diagram

The bio-metric fingerprint scanner will serve as the input for the computer through the serial port of the circuit that will communicate with the computer. Max 232 will receive the address from the serial port. The address will then be passed to the microcontroller unit (MCU). After which the address will be compared with the address encoded on the MCU if the address matched. A signal will then be sent to the relay and through the relay the solenoid will lock or unlock. The Direct Current (DC) motor will then rotate to open or close.

As shown in Figure 4, a 220 input AC voltage was supplied to the circuit. This voltage was transformed and lowered to power up the MAX 232, MCU, relay, the solenoid and the DC motor components.

Schematic Diagram

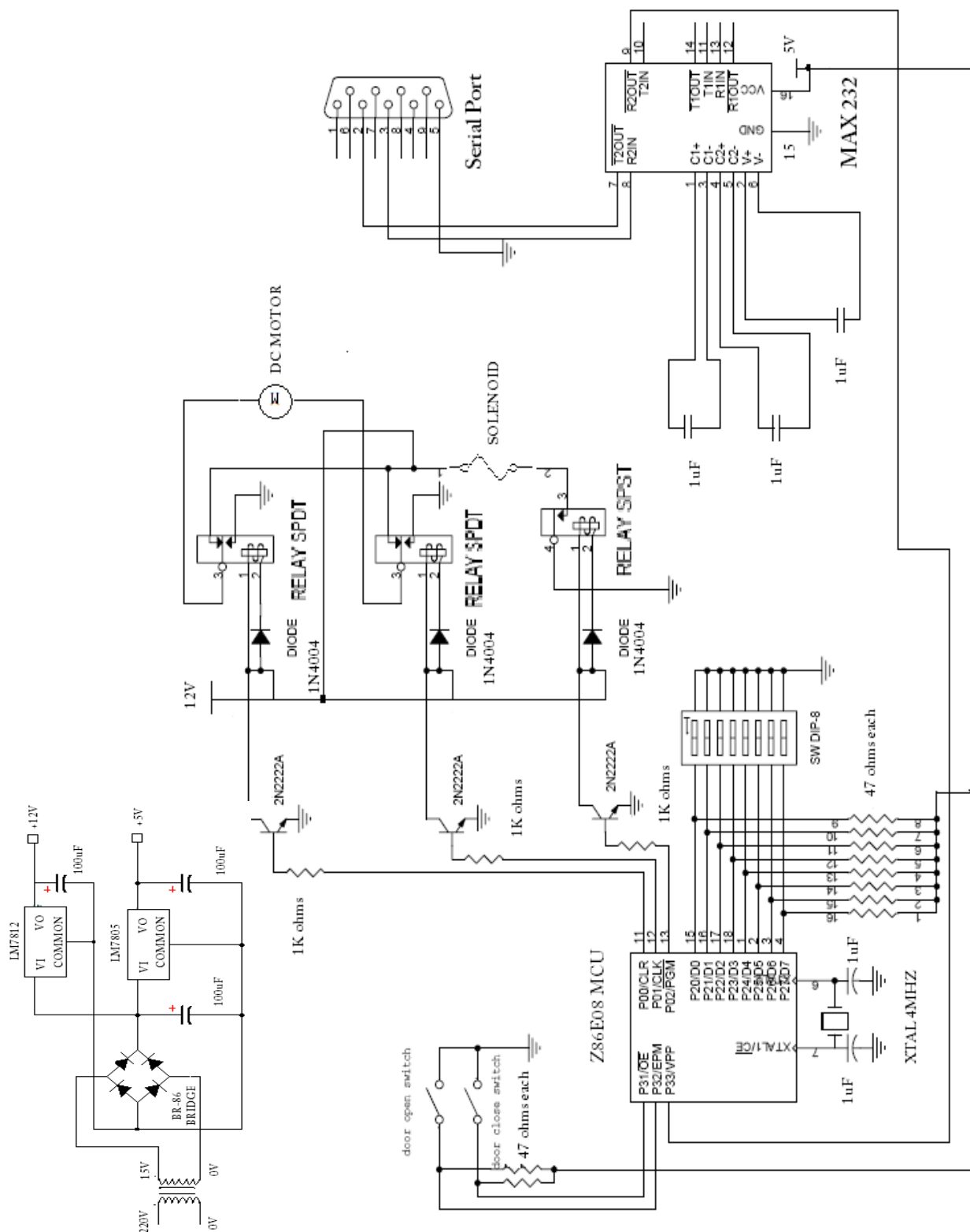


Figure 5 – Door Lock / Unlock Trigger Circuit

The circuit illustrated in Figure 5 is connected to the computer terminal through the serial port. The MAX 232 is needed for this condition. MAX 232 converts the port signal to a Transistor-Transistor Logic/Complementary Metal Oxide Semiconductor level pulse. Since the circuit only receives signals from the serial port R1 is not connected to the circuit. RS232 output is then connected to Port 3 of the microcontroller. Port 0 of the microcontroller serves as the output for the system which is connected to the relay module. Port 2 is used for the Dip switch. This switch however is not vital to the system. This is placed for testing the address only. Port 2 is connected to the toggle switch for controlling the voltage input to the microcontroller. If the switch is turned off then the circuit will not react even if MAX 232 receives a data from the serial port.

Hardware Components

The hardware components are also divided into three categories, namely: input, process, and output. MAX 232 is considered as the input for this system since it receives the signal from the serial port to the microcontroller. It converts the RS 232 level to a TTL/CMOS level which is needed by the microcontroller to be understood. This address is sent to the microcontroller. The microcontroller will determine if this address is valid to trigger the relay to open or close the door. If there is a match of address then the relay will be triggered to control the output of the system. The relay will latch or toggle the state of the output devices. The system takes two outputs, one for the solenoid and another for the DC motor.

1. Component Name: Microcontroller

The microcontroller is in charge of receiving the decimal data from the max 232 and converts it into hexadecimal code for the address of the door. It also sends signal to the relay to control the solenoid and the DC motor.

The Zilog's Z86E08 Microcontrollers (MCU) is used. Port 0 is dedicated to the output of the circuit and connected to the relay which is responsible for latching the state of the solenoid and the DC motor. Port 2 is used for changing the address contained in the microcontroller. This is the basis of the microcontroller whether to open or close the door. If the input address coming from this port is equal to the address received from Port 33 of the microcontroller, a triggering of the relay will result. Ports 31 and 32 on the other hand, serves as the power switch for the microcontroller.

2. Component Name: MAX 232

This is a receiver that converts a serial input from RS 232 to 5V TTL /CMOS level for the microcontroller. The R2 input of MAC 232 receives the data from the serial port while its R2 output is connected to Port 33 of the microcontroller. This component is vital to the circuit since the circuit is interfaced to a computer serial port.

3. Component Name: Solenoid

The solenoid is used as a lock for the door in the system. The solenoid is considered as an output for the circuit since it only waits signal from the relay to latch its state. It is operating on 12V input.

4. Component Name: Relay Module

The relay is used for triggering the state of the devices. It is connected to some of the ports of the microcontroller specifically to Port 0 of the Z86E08 MCU. The

microcontroller sends signal to the relay to activate the device connected. The module consists of a 12V relay and 3-pins terminal block. The module was chosen mainly because of availability. This will trigger the DC motor and the solenoid. Its task is to latch or toggle the state of the output.

5. Component Name: Power Supply

The power supply is used to provide the devices in the system with proper voltage requirements. The transformer is used to drop the input voltage. The rectifier, together with the capacitor converts the AC signal into DC. The voltage regulator then filters the voltage to 5V to supply the microcontroller and MAX 232 and 12V for the solenoid, relays and DC motor. The power supply consists of a transformer, a bridge-type rectifier and voltage regulator. The transformer used was a step-down transformer with a 12V output chosen mainly to fit the corresponding input voltage supply needed by the voltage regulator and the relay module.

6. Component Name: Toggle Switch

The SPDT serves as a power switch for the microcontroller. This switch is needed to restart the MCU in the circuit, if ever a malfunction occurs in the system.

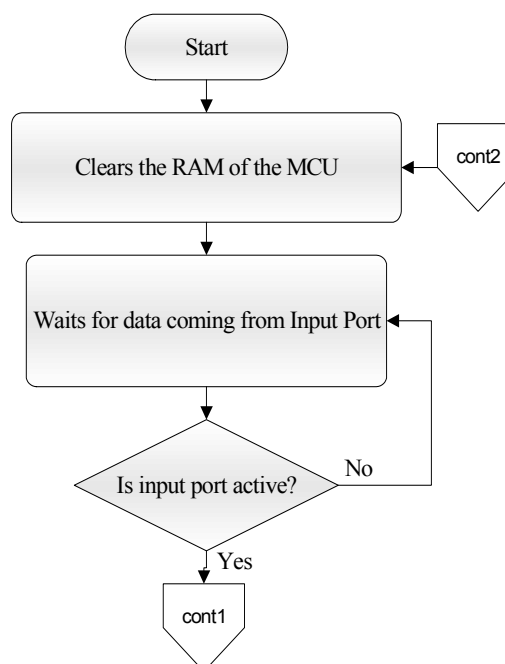
7. Component Name: DC motor

The DC motor is used to control the movement of the door. The motor rotates 360 degrees, so limit switches are placed below the door to prevent it from rotating over 90 degrees.

Hardware Implementation

Based from the hardware components discussed in the previous pages of this research, most of the materials can be used in the actual design of the circuit. Since a miniature DC stepper motor is used for the prototype, different motors can be used to replace it. Actually an AC motor can replace the DC motor however, DC motors has more accurate speed, is faster, more efficient and have position control. The user can also disregard the use of motor since the motor's purpose is only to rotate the door for a certain angle. The solenoid can be replaced with a linear solenoid operated door lock with DC type spring return. For the actual placement of the fingerprint scanner it should be placed at the corner outside the door. The power supply should be placed on top of the door near the motor for safety. The computer terminal should be located inside the administrator office near the door. If this is not possible a USB self powered hub can be used.

System Flowchart



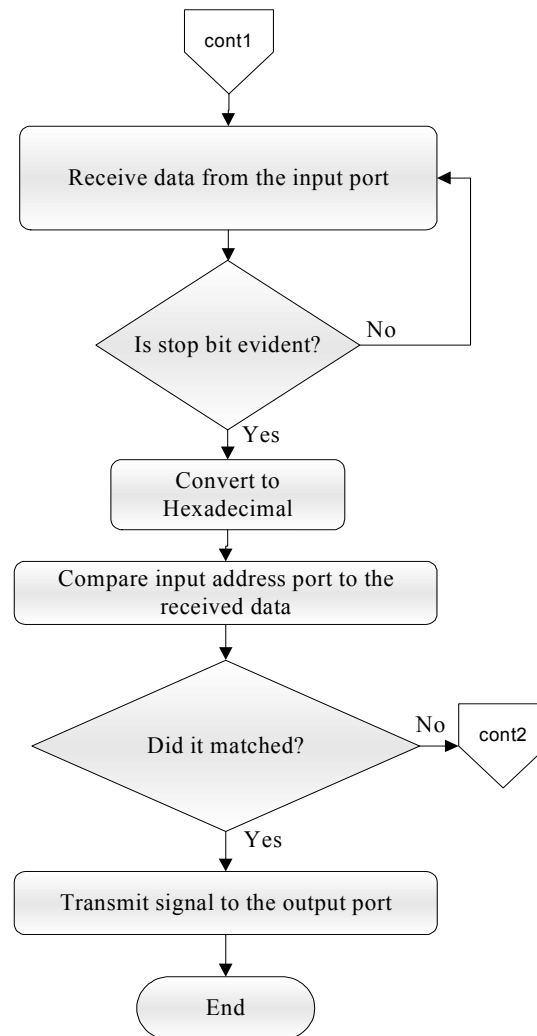


Figure 6 – Hardware System Flowchart

To discuss the flow of the hardware program a diagrams shown in Figure 6. First it will clear the memory of the MCU then waits for the received data. After receiving the data, it will then be converted to the data from decimal to hexadecimal format as the data coming from the software is in decimal format, so a routine is included in the program to convert data format. After conversion, it is then compared to the bits coming from Ports 20 to 27, if it matches then it will transmit a signal to Ports 00 to 02. Port 33 and Port 2 (address port) which serves as the input ports for the microcontroller while Port 0 for the output of the microcontroller.

SOFTWARE DESIGN

This software design provides a description of the design of the application software for automated fingerprint activated door lock (AFPADL) which was developed by the BIOLOCK researchers' design. The dominant design methodology is an object-oriented design using a visual interface to a database management system.

One part of the system is made for activating the door enabling it to lock and unlock. This part requires a finger reading for the authentication of the access to the room. Another part of the system is available for the administrators' control over the design. This function is capable of adding, deleting and updating the records and schedules of the persons who can access the room.

The user will do most normal maintenance of the persistent data in the database using database utilities. These include adding and deleting of professors' profile and their fingerprints, editing or updating schedules and monitoring the daily access to the room.

The user has access to this system through forms. These forms interact with several code modules to provide the bulk of the services. In turn these code modules interact with the underlying database.

This system is designed to run only in 1 computer terminal. It is a stand-alone application which does not require any internet connection or networking capabilities but needs a serial port or a USB to serial port plug. For a detailed discussion of the software application please refer to Appendix F.

System Flowchart

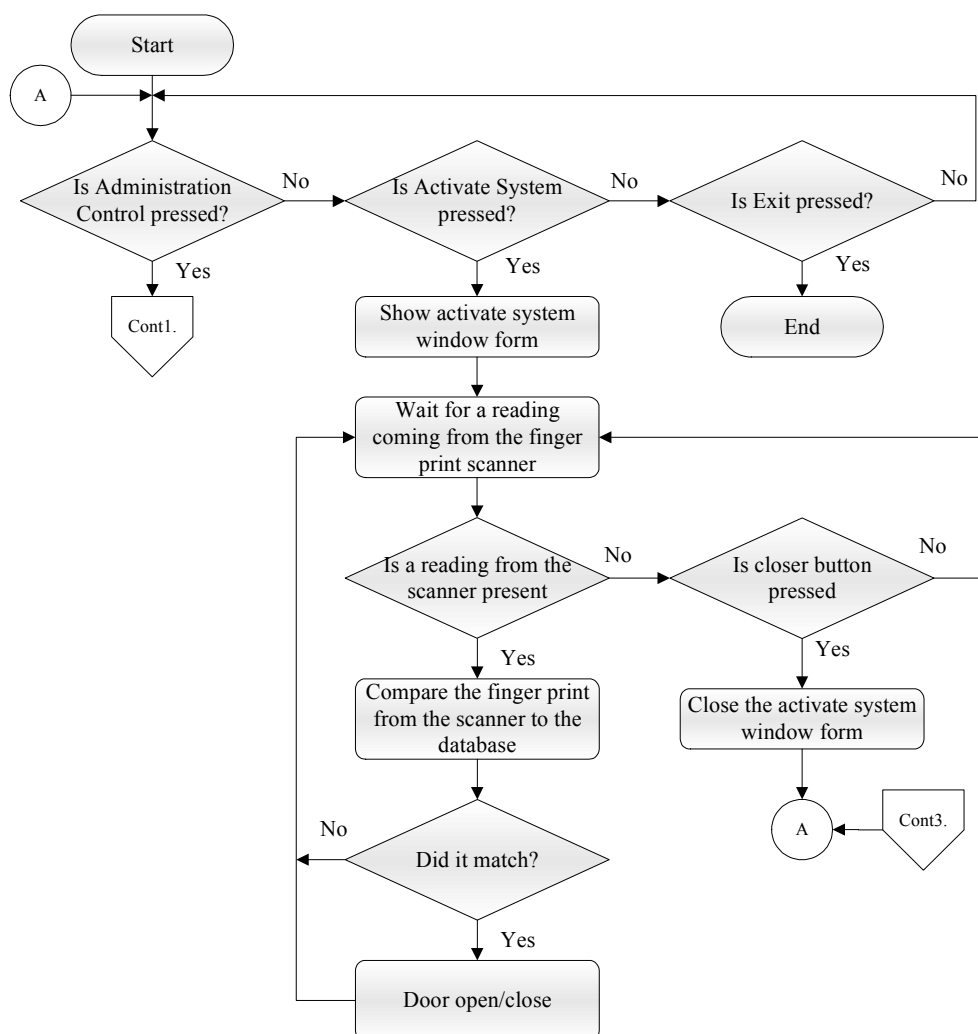


Figure 7 – Main Menu Activate System Flowchart

The software application is capable of locking and unlocking the door by sending an address through the serial port to the circuit. The software is created to check whether a reading from the fingerprint scanner matched the fingerprint binary data stored in the database. It also handles the comparison of the current system time of placing a finger to the bio-metric scanner from the schedule hard code in the program. These are the two conditions that should be fulfilled to be able to lock or unlock the door. To understand the details on how the software works a flowchart is illustrated in Figure 7.

Figure 7 also shows the main menu functions of the software application, as well as the detailed flow for activating the system. Once the activate system button is pressed, it will be disabled and can only become enabled after closing the activate system window form.

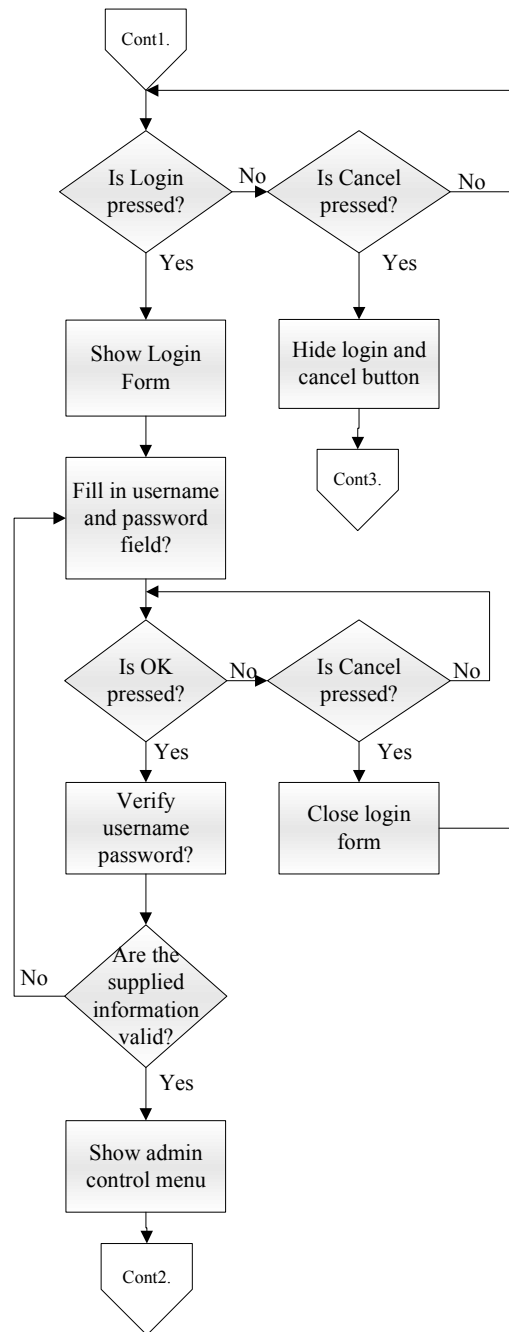


Figure 8 – Login Flowchart

The application software is vital to the system. Since security is considered in the design, a log-in procedure is added to the system as shown in Figure 8. Passwords are encrypted to prevent database hackers from accessing the system.

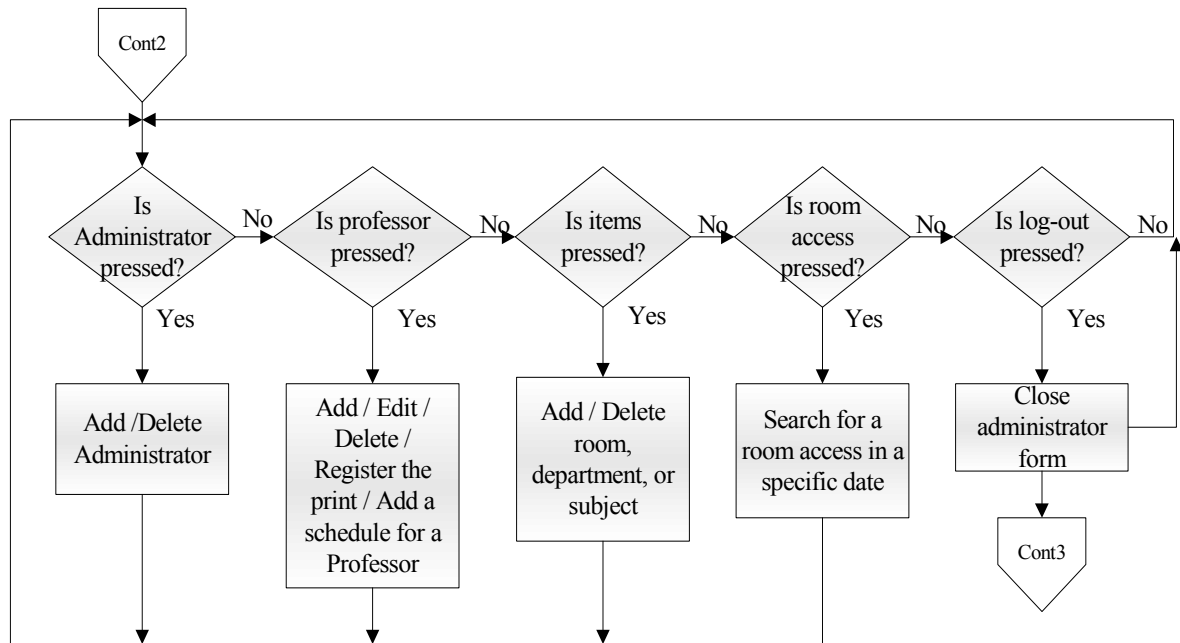


Figure 9 – Administration Control Flowchart

After a successful log-in, administrator controls are now enabled. Referring to Figure 9, the administrator can add or delete another administrator and other features needed for accessing a room.

For the database of the software, an entity-relationship diagram is presented to explain the contents of the database.

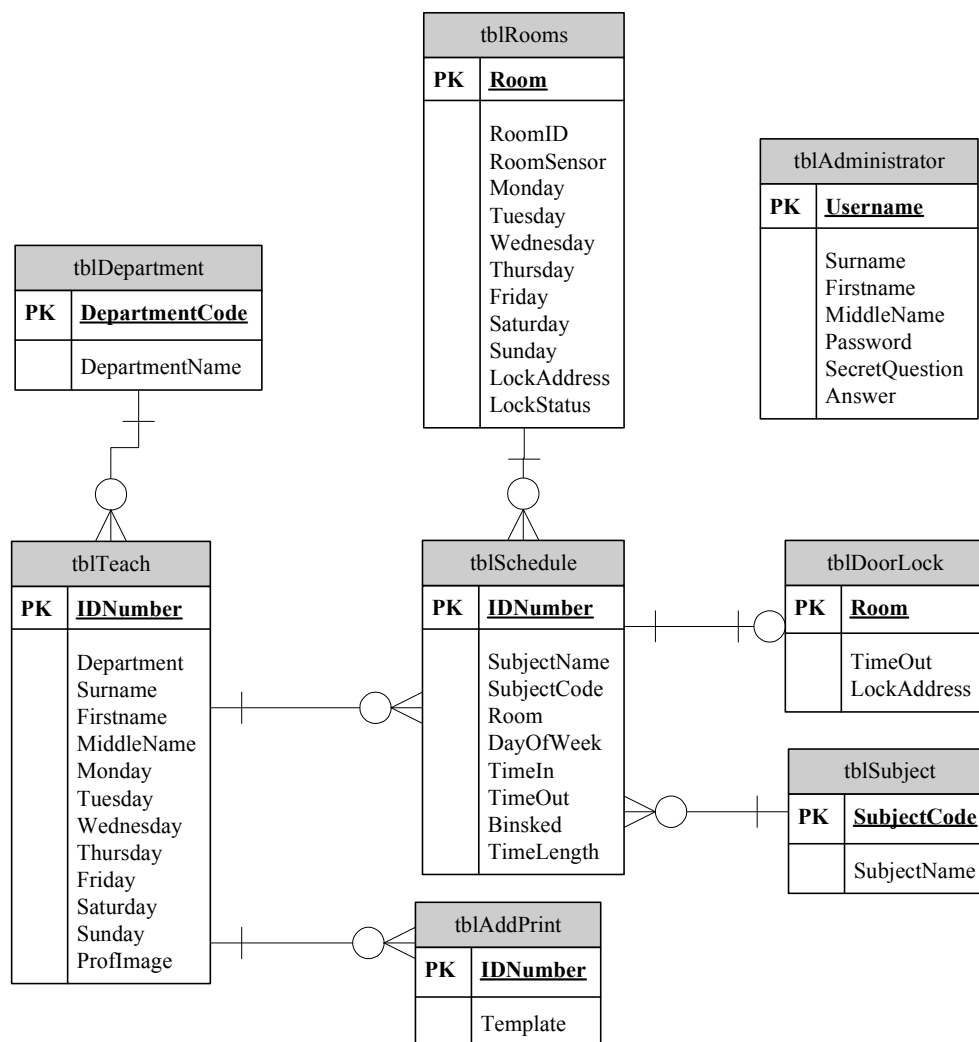


Figure 10 – Database Entity-Relationship Diagram

Referring to Figure 10, the database table **tblAdministrator** does not have any relationship with the other tables because it is only used as reference by the administrators. The **tblTeach** is the table for the professors. The relationship of **tblTeach** to **tblAddprint**, which is the table of record for fingerprints, is “one is to zero or more” because a professor can have multiple fingerprint templates or none at all. The same goes for the relationship between the **tblteach** and **tblSchedule** which is a table for all of the professor’s schedules. The relationship of **tblDepartment** with **tblTeach** is also “one is to

zero or more” because it will determine which department the professor belongs. The tblRooms is the records of the rooms that have the bio-metric locks. Its relationship with tblSchedule is “one to zero or many” because a room can have several schedules as long as they do not overlap with each other or the room can be vacant and not have any schedules at all. The tblSchedule has a relationship with tblDoorLock which is a “one to zero or one” because one schedule can only be connected to tblDoorLock or none at all. The tblDoorLock is used to store the rooms that are being occupied by the professors. Lastly, tblSubject has a “one to zero or many” relationship with tblSchedule because a certain subject can have several schedules or none at all.

Chapter 3

PRESENTATION AND INTERPRETATION OF DATA

Accuracy of the Design

In testing the accuracy of the design different trials were made. To test the functionality of the system a number of trials were made. All trials were done when the door was initially closed. Table 2 shows the trials made by a user (professor) that has a record of fingerprints and scheduled on Monday 7:00 am to 7:05 am. The testing was done on the same scheduled day with different time in and time out. An okay mark for the record means the user's fingerprint was present in the database. An x suggested otherwise.

Trials	Time		Scheduled (Monday)		Record (fingerprint)	Results
	in	out	In	Out	ok / x	Open, Close door
1	07:00 AM	07:02 AM	07:00 AM	07:05 AM	ok	open, close
2	07:00 AM	07:05 AM	07:00 AM	07:05 AM	ok	open, close
3	07:03 AM	-	07:00 AM	07:05 AM	ok	-
4	07:01 AM	07:03 AM	07:00 AM	07:05 AM	ok	open, close
5	07:00 AM	07:03 AM	07:00 AM	07:05 AM	ok	open, close
6	07:00 AM	07:04 AM	07:00 AM	07:05 AM	ok	open, close
7	07:10 AM	-	07:00 AM	07:05 AM	ok	-
8	07:06 AM	-	07:00 AM	07:05 AM	ok	-
9	07:20 AM	-	07:00 AM	07:05 AM	ok	-
10	07:00 AM	-	07:00 AM	07:05 AM	ok	open, close

Table 2 – First Trial Table

The first trial shown in Table 2 was done by a user that had fingerprint records and had a scheduled access to the room. In this table, time in refers to the time he placed his finger on the scanner. Users who are valid to enter a room can open and close the door within the grace period of two minutes. After that period, a user can no longer open or unlock the door. The user, however, can only lock the door if it is left open through a swipe of his/her fingerprint to the scanner. If the user is not scheduled for that time even if a record of his/her fingerprint existed, s/he cannot enter the door as shown in trials 7-9.

	Time		Scheduled (Monday)		Record (fingerprint)	Results
Trials	in	Out	In	out	ok / x	Open, Close door
1	07:00 AM	-	07:00 AM	07:05 AM	X	-
2	07:01 AM	-	07:00 AM	07:05 AM	X	-
3	07:02 AM	-	07:00 AM	07:05 AM	X	-
4	07:03 AM	-	07:00 AM	07:05 AM	X	-
5	07:04 AM	-	07:00 AM	07:05 AM	X	-
6	07:05 AM	-	07:00 AM	07:05 AM	X	-
7	07:06 AM	-	07:00 AM	07:05 AM	X	-
8	07:10 AM	-	07:00 AM	07:05 AM	X	-
9	07:11 AM	-	07:00 AM	07:05 AM	X	-
10	07:12 AM	-	07:00 AM	07:05 AM	X	-

Table 3 – Second Trial Table

Referring to Table 3, on the other hand, a user has no fingerprint record and is scheduled on Monday 07:00 am to 07:05 am. In any time of the day, even if a user is

given a schedule in using the room, s/he cannot use the room or will not be able to open or close the door without a fingerprint record since fingerprints are used to access the room.

	Time		Scheduled (Monday)		Record (fingerprint)	Results
Trials	In	Day	in	out	ok / x	Open door
1	07:00 AM	Monday	07:00 AM	07:05 AM	ok	open
2	07:01 AM	Tuesday	07:00 AM	07:05 AM	ok	-
3	07:02 AM	Wednesday	07:00 AM	07:05 AM	ok	-
4	07:03 AM	Thursday	07:00 AM	07:05 AM	ok	-
5	07:04 AM	Friday	07:00 AM	07:05 AM	ok	-
6	07:05 AM	Monday	07:00 AM	07:05 AM	ok	-
7	07:06 AM	Sunday	07:00 AM	07:05 AM	ok	-
8	07:10 AM	Monday	07:00 AM	07:05 AM	ok	-
9	07:11 AM	Saturday	07:00 AM	07:05 AM	ok	-
10	07:12 AM	Monday	07:00 AM	07:05 AM	ok	-

Table 4 – Third Trial Table

Access to the room was done on different days of the week and on different time as shown in Table 4. The user can only access a room for his/her given scheduled time and day. Since the user is only scheduled on Monday from 7:00 am to 7:05 am, s/he cannot have access to a room on any other day and time.

	Time		Scheduled (Monday)		Record (fingerprint)	Results
Trials	In	Day	in	out	ok / x	Open door
1	07:00 AM	Monday	-	-	-	-
2	07:01 AM	Tuesday	-	-	-	-
3	07:02 AM	Wednesday	-	-	-	-
4	07:03 AM	Thursday	-	-	-	-
5	07:04 AM	Friday	-	-	-	-
6	07:05 AM	Monday	-	-	-	-
7	07:06 AM	Sunday	-	-	-	-
8	07:10 AM	Monday	-	-	-	-
9	07:11 AM	Saturday	-	-	-	-
10	07:12 AM	Monday	-	-	-	-

Table 5 – Fourth Trial Table

For Table 5, the user has no record and not scheduled for any day and time which means that the user has no authority to have access to the room. The table showed how a schedule to access a room and a fingerprint record is significant to the system. Based on the previous discussions since these two variables are inputs to the system, without both, users cannot access a room.

Based on the trials made, the door actually opens and closes if a record of finger print is present in the database and the user has a scheduled time for that specific time and day. After the grace period in Table 1, Trial 3 even if the user is scheduled for 7:00 to 7:05 am, the user was not able to open the door. This shows that the system functions and works based from the conditions stated earlier in the document.

Trials	Identified / not identified	Quality
1	Identified	High
2	Identified	Medium
3	Identified	Medium
4	Identified	High
5	Not Identified	Low
6	Identified	High
7	Identified	Medium
8	Identified	High
9	Identified	High
10	Not Identified	Low

Table 6 – Biometric Device Trial

Another testing was done to the input device which is the fingerprint scanner. This will test whether the device can identify or recognize a fingerprint as shown in Table 6 that a user has a fingerprint record.

Out of 10 trials, 8 fingerprint readings were identified. Based on Table 6, it showed that the devices may require a user for multiple fingerprint readings since it showed that not all trials were successful. Take note that the trials in the said table make a slight different angle change upon placing the finger on the scanner.

Chapter 4

CONCLUSION AND RECOMMENDATION

Conclusion

Based on the results from the testing, the system has achieved its objective to lock and unlock a door through the use of biometric device/fingerprint scanner interfaced with an application program. The system does its functionality accurately which is to lock and unlock the door at certain time interval. With this functionality the problem can be solved. Through the use of this design, people will have an easy way of having a convenient, secured and authorized entrance in a certain room. There would be no use for keys, passwords and cards. By just registering the fingerprint of a user and giving him/her a specific schedule, the door can be opened by the registered fingerprint and will automatically close after the specified time and open at a certain time interval. This automation can help people, security guards and utility men, administrators and professors in particular, which now has the easiest and secured way of accessing a certain room in a routinary based schedule. The good results that it brings are: less time in locking or unlocking the door because the design will use fingerprints to access the room which implies that the system is efficient. The room is more secured because of the unique fingerprint readings that the bio-metric device offers. It saves time, effort and replaces the use of keys, passwords and cards that often may be lost or forgotten.

Recommendation

It is recommended that the software should be modified into a personnel attendance monitoring device. It is also recommended to be used in schools and offices to provide more secure environment and to enforce authorized entry. This should be implemented to avoid borrowing of identity during enrolment or other vital activities in school and offices.

An uninterruptible power supply (UPS) can be added to the computer and the circuits power supply so when there is power cut off, there will still be temporary electric supply for the system to function smoothly. From these recommendations there would be no hassle in using the system.

For future developments, the system can further be modified depending on the clients' request and application. A combined additional biometric device is recommended for more secured access to rooms.

BIBLIOGRAPHY

Araneta, A., et. al., PIC-based Doorlock System, September 2003.

Badrkhan K.S., et. al., Electronics: Principles and Application, Southwestern, Cincinnati, Ohio, 1984.

Burgess, Peter, The State of Biometric Industry: The Search for Security and Convenience, July 2001.

Guverich V., Electrical Relays: Principles and Applications, CRC Press Taylor and Francis, New York, 2005.

Nielsen, Paul. SQL Server 2005 Bible, Indianapolis, Indiana: Wiley Publishing Inc., 2007.

Price, Anne. Murach's Beginning Visual Basic .NET, United States of America: Mike Murach & Associates Inc., 2002.

Sempf, Bill. Visual Basic 2005 For Dummies, Indianapolis, Indiana: Wiley Publishing Inc., 2006.

Viscusi, S., Sequiam Biometric Door Lock, 2006.

APPENDIX A
LIST OF MATERIALS

List of Materials for the Hardware

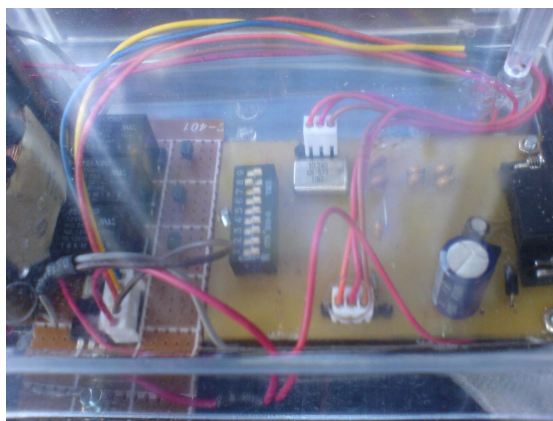
Components	Quantity
Max 232	1pc
Z86E08 MCU	1pc
4 MHz crystal oscillator	1pc
Dip switch 8-bit	1pc
PCB 3x3"	1pc
Relay SPDT	3pcs
Sockets & AC cord	1pc
Transformer (220Vac input/ 12Vac/5Vac output)	1pc
Solenoid	1pc
DC stepper geared motor	1pc
1 uf Capacitor	6pcs
Serial cable	1pc
Acrylic 1f ²	1pc
Transistor NPN	3pcs
Toggle switch	1pc
Resistors	14pcs

APPENDIX B
PICTURES OF THE DESIGN

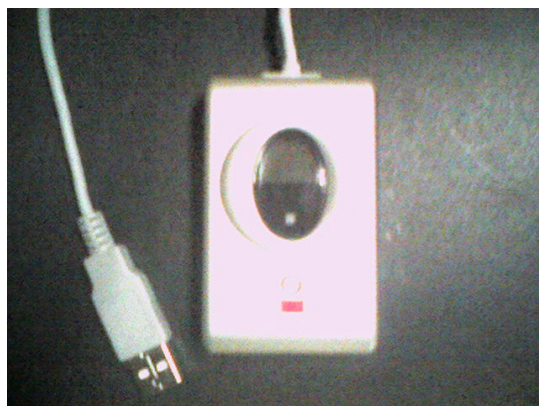
Pictures of the Design



Front View of the System Hardware



Top View of the System Hardware (Circuit)



Picture of the Biometric Lock

*Please refer to Appendix F for the Pictures of the System Software

APPENDIX C
SOURCE CODE

DBClass.vb

```
Imports System.Runtime.InteropServices
Imports System.Data.SqlClient

' Template data
Public Class TTemplate
    ' Template itself
    Public tpt As System.Array =
        Array.CreateInstance(GetType(Byte),
            GrFingerXLib.GRConstants.GR_MAX_SIZ
            E_TEMPLATE)
    ' Template size
    Public Size As Long
End Class
' Template list
Public Structure TTemplates
    ' ID
    Public ID As String
    ' Template itself
    Public template As TTemplate
End Structure

Public Class DBClass
    Dim conn As SqlConnection
    ' Open connection
    Public Function OpenDB() As Boolean
        Dim test As New getconnstring
        conn = New SqlConnection(test.getConn)
        Try
            conn.Open()
            Return True
        Catch
            Return False
        End Try
    End Function
    ' Close connection
    Public Sub closeDB()
        conn.Close()
    End Sub
    ' Add template to database. Returns added template
    ID.
    Public Function AddTemplate(ByRef template As
    TTemplate) As Long
        Dim da As New SqlDataAdapter("select * from
tblAddPrint", conn)
        Dim dt As DataTable
        conn.Open()
        dt = New DataTable
        da.Fill(dt)
        cmb = New SqlCommandBuilder(da)
        newRow = dt.NewRow
        newRow("IDNumber") = profid
        newRow("template") = template.tpt
        dt.Rows.Add(newRow)
        da.InsertCommand = cmb.GetInsertCommand
        da.Update(dt)
        conn.Close()
        ' return ID
        Return newRow("IDNumber")
    End Function
    ' Returns a DataTable with all enrolled templates
    from database.
    Public Function getTemplates() As TTemplates()
        Dim ds As New DataSet
        Dim da As New SqlDataAdapter("select * from
tblAddPrint", conn)
        Dim ttpts As TTemplates()
```

```
Dim i As Integer
' Get query response
da.Fill(ds)
Dim tpts As DataRowCollection =
ds.Tables(0).Rows
' Create response array
ReDim ttpts(tpts.Count)
' No results?
If tpts.Count = 0 Then Return ttpts
' get each template and put results in our array
For i = 1 To tpts.Count
    ttpts(i).template = New TTemplate
    ttpts(i).ID = tpts.Item(i - 1).Item("IDNumber")
    ttpts(i).template.tpt = tpts.Item(i -
1).Item("template")
    ttpts(i).template.Size =
ttpts(i).template.tpt.Length
Next
Return ttpts
End Function
' Returns template with supplied ID.
Public Function getTemplate(ByVal id As Long) As
Byte()
    Dim ds As New DataSet
    Dim da As New SqlDataAdapter("Select * from
tblAddPrint where IDNumber = " &
    profid, conn)
    Dim tpt As New TTemplate
    ' Get query response
    da.Fill(ds)
    Dim tpts As DataRowCollection =
ds.Tables(0).Rows
    ' No results?
    If tpts.Count <> 1 Then Return Nothing
    ' Deserialize template and return it
    Return tpts.Item(0).Item("template")
End Function
End Class
encrypt.vb
```

```
Imports System.Security.Cryptography

Public Class encrypt
    Public NotInheritable Class Simple3Des
        Private TripleDes As New
        TripleDESCryptoServiceProvider

        Private Function TruncateHash(_
            ByVal key As String, _
            ByVal length As Integer) _
            As Byte()

            Dim sha1 As New SHA1CryptoServiceProvider

            ' Hash the key.
            Dim keyBytes() As Byte = _

System.Text.Encoding.Unicode.GetBytes(key)
            Dim hash() As Byte =
sha1.ComputeHash(keyBytes)

            ' Truncate or pad the hash.
            ReDim Preserve hash(length - 1)
            Return hash
        End Function

        Sub New(ByVal key As String)
            ' Initialize the crypto provider.
```

```

        TripleDes.Key = TruncateHash(key,
TripleDes.KeySize \ 8)
        TripleDes.IV = TruncateHash("",
TripleDes.BlockSize \ 8)
    End Sub

    Public Function EncryptData( _
        ByVal plaintext As String) _
        As String

        ' Convert the plaintext string to a byte array.
        Dim plaintextBytes() As Byte = _
System.Text.Encoding.Unicode.GetBytes(plaintext)

        ' Create the stream.
        Dim ms As New System.IO.MemoryStream
        ' Create the encoder to write to the stream.
        Dim encStream As New CryptoStream(ms, _
            TripleDes.CreateEncryptor(), _
System.Security.Cryptography.CryptoStreamMode.Write)
        ' Use the crypto stream to write the byte array to
the stream.
        encStream.Write(plaintextBytes, 0,
plaintextBytes.Length)
        encStream.FlushFinalBlock()
        ' Convert the encrypted stream to a printable
string.
        Return Convert.ToBase64String(ms.ToArray)
    End Function
End Class
End Class

```

frmAddprint.vb

```

Imports GrFingerXLib
Imports Microsoft.VisualBasic

Public Class frmAddPrint
    Inherits System.Windows.Forms.Form
    Dim myUtil As Util

    Private Sub MainForm_Load(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
        Handles MyBase.Load
        Dim err As Integer
        btnAdd.Enabled = False
        ' initialize util class
        myUtil = New Util(LogList, PictureBox1,
AxGrFingerXCtrl1)
        ' Initialize GrFingerX Library
        err = myUtil.InitializeGrFinger()
        ' Print result in log
        If err < 0 Then
            myUtil.WriteError(err)
        Exit Sub
        Else
            myUtil.WriteLog("**GrFingerX Initialized
Successful**")
        End If
    End Sub

    Private Sub MainForm_Close(ByVal sender As
        System.Object, ByVal e As
        System.ComponentModel.CancelEventArgs)
        Handles MyBase.Closing
        myUtil.FinalizeGrFinger()
    End Sub

```

```

    Private Sub btnAdd_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
        Handles btnAdd.Click
        Dim id As Integer
        ' add fingerprint
        id = myUtil.Enroll()
        ' write result to log
        If id >= 0 Then
            myUtil.WriteLog("Fingerprint enrolled with id
= " & id)
        Else
            myUtil.WriteLog("Error: Fingerprint not
enrolled")
        End If
    End Sub

    ' A fingerprint reader was plugged on system
    Private Sub AxGrFingerXCtrl1_SensorPlug(ByVal sender
        As System.Object, ByVal e As
        AxGrFingerXLib._IGrFingerXCtrlEvents_Sensor
        PlugEvent) Handles
        AxGrFingerXCtrl1.SensorPlug
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Plugged.")
        AxGrFingerXCtrl1.CapStartCapture(e.idSensor)
    End Sub ' A fingerprint reader was unplugged from
system
    Private Sub AxGrFingerXCtrl1_SensorUnplug(ByVal
        sender As System.Object, ByVal e As
        AxGrFingerXLib._IGrFingerXCtrlEvents_SensorU
        nplugEvent) Handles
        AxGrFingerXCtrl1.SensorUnplug
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Unplugged.")
        AxGrFingerXCtrl1.CapStopCapture(e.idSensor)
    End Sub

    ' A finger was placed on reader
    Private Sub AxGrFingerXCtrl1_FingerDown(ByVal
        sender As System.Object, ByVal e As
        AxGrFingerXLib._IGrFingerXCtrlEvents_Finger
        DownEvent) Handles
        AxGrFingerXCtrl1.FingerDown
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Finger Placed.")
    End Sub

    ' A finger was removed from reader
    Private Sub AxGrFingerXCtrl1_FingerUp(ByVal sender
        As System.Object, ByVal e As
        AxGrFingerXLib._IGrFingerXCtrlEvents_Finger
        UpEvent) Handles AxGrFingerXCtrl1.FingerUp
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Finger removed.")
    End Sub

    ' An image was acquired from reader
    Private Sub AxGrFingerXCtrl1_ImageAcquired(ByVal
        sender As System.Object, ByVal e As
        AxGrFingerXLib._IGrFingerXCtrlEvents_Image
        AcquiredEvent) Handles
        AxGrFingerXCtrl1.ImageAcquired
        ' Copying aquired image
        myUtil.raw.height = e.height
        myUtil.raw.width = e.width
        myUtil.raw.res = e.res
        myUtil.raw.img = e.rawImage
        ' Signaling that an Image Event occurred.

```

```

        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Image captured.")

        ' display fingerprint image
        myUtil.PrintBiometricDisplay(False,
GRConstants.GR_DEFAULT_CONTEXT)

        ' now we have a fingerprint, so we can extract
template
        Dim ret As Integer

        ' extract template
        ret = myUtil.ExtractTemplate()
        ' write template quality to log
        If ret = GRConstants.GR_BAD_QUALITY Then
            myUtil.WriteLog("Template extracted
successfully. Bad quality.")
        ElseIf ret =
GRConstants.GR_MEDIUM_QUALITY Then
            myUtil.WriteLog("Template extracted
successfully. Medium quality.")
        ElseIf ret = GRConstants.GR_HIGH_QUALITY
Then
            myUtil.WriteLog("Template extracted
successfully. High quality.")
        End If
        If ret >= 0 Then
            ' if no error, display
minutiae/segments/directions into the image
            myUtil.PrintBiometricDisplay(True,
GRConstants.GR_NO_CONTEXT)
        ' enable operations we can do over extracted
template
            btnAdd.Enabled = True
        Else
            ' write error to log
            myUtil.WriteError(ret)
        End If
    End Sub

    Private Sub btnExit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Handles btnExit.Click
        myUtil.FinalizeGrFinger()
        Me.Close()
    End Sub
End Class

```

frmAddProfessor.vb

```

Imports System.Data.SqlClient

Public Structure ExistingSched
    Public IDNumber As String
    Public Room As String
    Public DayOfWeek As String
    Public BinSked As String
End Structure

Public Class frmAddProfessor
    Dim conn As SqlConnection

    Private Sub frmAddProfessor_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Handles MyBase.Load
        Dim test As New getconnstring
        conn = New SqlConnection(test.getconn)
        Gridview()
        LoadDepartment()
        btnAddSchedule.Enabled = False
    End Sub

```

```

        btnRegisterPrint.Enabled = False
        btnDelete.Enabled = False
        btnDelete.Hide()
        btnCancel2.SendToBack()
    End Sub

    Private Sub LoadDepartment()
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select * from
tblDepartment", conn)
        conn.Open()
        da.Fill(ds, "department")
        cboDepartment.DataSource =
ds.Tables("department")
        cboDepartment.DisplayMember =
"DepartmentCode"
        conn.Close()
        cboDepartment.Show()
        cboDepartment.SelectedItem = 0
        txtDepartment.Hide()
    End Sub

    Private Sub Gridview()
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select
Surname,Firstname,Middlename,IDNumber from tblTeach",
conn)
        conn.Open()
        da.Fill(ds, "teach")
        grdAddProfessor.DataSource = ds.Tables("teach")
        conn.Close()
    End Sub

    Private Sub Clear()
        txtSurname.Text = Nothing
        txtFirstname.Text = Nothing
        txtMiddlename.Text = Nothing
        txtID.Text = Nothing
        cboDepartment.SelectedItem = 0
        txtDepartment.Text = Nothing
        btnAddSchedule.Enabled = False
        btnRegisterPrint.Enabled = False
    End Sub

    Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
        Handles btnAdd.Click

```

```

*****
*****
        'Check for Information Deficiency
*****
*****
        If txtID.Text.Trim = Nothing Then
            MessageBox.Show("The Identification Number Field
is empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        Exit Sub
        ElseIf txtSurname.Text.Trim = Nothing Then
            MessageBox.Show("The Surname field is empty",
"Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        Exit Sub
        ElseIf txtFirstName.Text.Trim = Nothing Then
            MessageBox.Show("The First Name field is empty",
"Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        Exit Sub
        ElseIf txtMiddleName.Text.Trim = Nothing Then
            MessageBox.Show("The Middle Name field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)

```

```

Exit Sub
End If
'*****
*****
'*****
*****
dt = New DataTable
da = New SqlDataAdapter("Select * from
tblTeach", conn)
conn.Open()
da.Fill(dt)
cmb = New SqlCommandBuilder(da)
'*****
*****
'Check for Redundancy
'*****
*****
cmd = New SqlCommand("Select IDNumber from
tblTeach where IDNumber = '" & txtID.Text.Trim
& "'", conn)
dr = cmd.ExecuteReader
If dr.Read Then
MessageBox.Show("Identification already exists in
the database", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
dr.Close()
conn.Close()
Exit Sub
Else
dr.Close()
End If
'*****
*****
'*****
*****
newRow = dt.NewRow
newRow("IDNumber") = txtID.Text.Trim
newRow("Surname") = txtSurname.Text.Trim
newRow("Firstname") = txtFirstName.Text.Trim
newRow("Middlename") =
txtMiddleName.Text.Trim
newRow("Department") =
cboDepartment.Text.Trim
newRow("Monday") = "00000"
newRow("Tuesday") = "00000"
newRow("Wednesday") = "00000"
newRow("Thursday") = "00000"
newRow("Friday") = "00000"
newRow("Saturday") = "00000"
newRow("Sunday") = "00000"
newRow("ProfImage") =
imgProfessor.ImageLocation
dt.Rows.Add(newRow)
da.InsertCommand = cmb.GetInsertCommand
da.Update(dt)
conn.Close()

btnAddSchedule.Enabled = False
btnRegisterPrint.Enabled = False
GridView()
Clear()
End Sub

Private Sub grdAddProfessor_CellClick(ByVal sender As
System.Object, ByVal e As System.EventArgs
System.Windows.Forms.DataGridViewCellEvent
Args) Handles grdAddProfessor.CellClick
btnAddSchedule.Enabled = True
btnRegisterPrint.Enabled = True
btnDelete.Enabled = True

```

```

btnCancel.BringToFront()
txtDepartment.BringToFront()

cmd = New SqlCommand("Select * from tblTeach
where IDNumber = '" & grdAddProfessor.Item(3,
grdAddProfessor.CurrentRow.Index).Value & "'",
conn)
conn.Open()
dr = cmd.ExecuteReader
If dr.Read Then
txtID.Text = dr("IDNumber")
txtDepartment.Text = dr("Department")
txtSurname.Text = dr("Surname")
txtFirstName.Text = dr("Firstname")
txtMiddleName.Text = dr("Middlename")
imgProfessor.SizeMode =
PictureBoxSizeMode.StretchImage
imgProfessor.ImageLocation = dr("ProfImage")

dr.Close()
conn.Close()
Else
dr.Close()
conn.Close()
End If
cboDepartment.Hide()
txtDepartment.Show()
btnCancel.Hide()
btnCancel2.Show()
btnCancel2.Enabled = True
btnCancel.Enabled = False
btnDelete.Show()
btnDelete.Enabled = True
End Sub

Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
Dim response As DialogResult
Dim RoomSked As String = Nothing
Dim NewRoomSked As String = Nothing
Dim index As String = Nothing

Dim ds As New DataSet
Dim da As New SqlDataAdapter("Select * from
tblSchedule where IDNumber = '" &
grdAddProfessor.Item(3,
grdAddProfessor.CurrentRow.Index).Value & "'",
conn)
conn.Open()
da.Fill(ds)
Dim Elist As ExistingSched()
Dim list As DataRowCollection =
ds.Tables(0).Rows
ReDim Elist(list.Count)

If list.Count <> 0 Then
For i As Integer = 1 To list.Count
Elist(i).IDNumber = list.Item(i -
1).Item("IDNumber")
Elist(i).Room = list.Item(i - 1).Item("Room")
Elist(i).DayOfWeek = list.Item(i -
1).Item("DayOfWeek")
Elist(i).BinSked = list.Item(i -
1).Item("DayOfWeek")
Next
conn.Close()
response = MessageBox.Show("The Selected
Professor has existing schedules. Are you sure
you want to delete him?", "Warning",

```

```

    MessageBoxButtons.YesNo,
    MessageBoxIcon.Exclamation)
    If response = Windows.Forms.DialogResult.Yes
Then
    For i As Integer = 1 To Elist.Length
        If i < Elist.Length Then
            cmd = New SqlCommand("Select * from
tblRooms where Room = '" &
Elist(i).Room & "'", conn)
            conn.Open()
            dr = cmd.ExecuteReader
            If dr.Read Then
                If Elist(i).DayOfWeek = "Monday"
Then
                    RoomSked = dr("Monday")
                    ElseIf Elist(i).DayOfWeek =
"Tuesday" Then
                        RoomSked = dr("Tuesday")
                    ElseIf Elist(i).DayOfWeek =
"Wednesday" Then
                        RoomSked = dr("Wednesday")
                    ElseIf Elist(i).DayOfWeek =
"Thursday" Then
                        RoomSked = dr("Thursday")
                    ElseIf Elist(i).DayOfWeek = "Friday"
Then
                        RoomSked = dr("Friday")
                    ElseIf Elist(i).DayOfWeek =
"Saturday" Then
                        RoomSked = dr("Saturday")
                    ElseIf Elist(i).DayOfWeek =
"Sunday" Then
                        RoomSked = dr("Sunday")
                    End If
                Else
                    dr.Close()
                End If
                conn.Close()
                '*****
                '*****
                'Get the New Room Schedule
                '*****
                '*****
                For j As Integer = 0 To 4
                    If RoomSked(j) = "1" And
Elist(i).BinSked(j) = "1" Then
                        index = "0"
                        NewRoomSked = NewRoomSked
+ index
                    Else
                        index = RoomSked(j)
                        NewRoomSked = NewRoomSked
+ index
                    End If
                Next
                '*****
                '*****
                '*****
                'Update the Room Binary Schedule
                Depending on the Day
                '*****
                '*****
                If Elist(i).DayOfWeek = "Monday"
Then
                    cmd = New SqlCommand("Update
tblRooms Set Monday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)

```

```

conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf Elist(i).DayOfWeek = "Tuesday"
Then
    cmd = New SqlCommand("Update
tblRooms Set Tuesday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)
    conn.Open()
    cmd.ExecuteNonQuery()
    conn.Close()
    ElseIf Elist(i).DayOfWeek =
"Wednesday" Then
        cmd = New SqlCommand("Update
tblRooms Set Wednesday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)
        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()
        ElseIf Elist(i).DayOfWeek = "Thursday"
Then
            cmd = New SqlCommand("Update
tblRooms Set Thursday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)
            conn.Open()
            cmd.ExecuteNonQuery()
            conn.Close()
            ElseIf Elist(i).DayOfWeek = "Friday"
Then
                cmd = New SqlCommand("Update
tblRooms Set Friday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)
                conn.Open()
                cmd.ExecuteNonQuery()
                conn.Close()
                ElseIf Elist(i).DayOfWeek = "Saturday"
Then
                    cmd = New SqlCommand("Update
tblRooms Set Saturday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)
                    conn.Open()
                    cmd.ExecuteNonQuery()
                    conn.Close()
                    ElseIf Elist(i).DayOfWeek = "Sunday"
Then
                        cmd = New SqlCommand("Update
tblRooms Set Sunday = '" &
NewRoomSked & "' where
Room = '" & Elist(i).Room & "'",
conn)
                        conn.Open()
                        cmd.ExecuteNonQuery()
                        conn.Close()
                        End If
                        '*****
                        '*****
                        '*****
                        'Delete the Schedule

```

```

*****
*****
cmd = New SqlCommand("Delete from
tblSchedule where Room = '" _
& Elist(i).Room & "' AND IDNumber =
'" _
& Elist(i).IDNumber & "' AND
DayOfWeek = '" _
& Elist(i).DayOfWeek & "' AND
BinSked = '" _
& Elist(i).BinSked & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
*****
*****
*****
*****
End If
Next
*****
*****
'Delete the Selected Professor
*****
*****
cmd = New SqlCommand("Delete from tblTeach
where IDNumber = '" &
grdAddProfessor.Item(3,
grdAddProfessor.CurrentRow.Index).
Value & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
*****
*****
*****
ElseIf response =
Windows.Forms.DialogResult.No Then
Exit Sub
End If
ElseIf list.Count = 0 Then
conn.Close()
*****
*****
'Delete the Selected Professor
*****
*****
*****
cmd = New SqlCommand("Delete from tblTeach
where IDNumber = '" &
grdAddProfessor.Item(3,
grdAddProfessor.CurrentRow.Index).
Value & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
*****
*****
*****
Exit Sub
Else
If MessageBox.Show("Are you sure you want to
delete the selected professor?",
"Warning!",
MessageBoxButtons.YesNo,
MessageBoxIcon.Information) =
Windows.Forms.DialogResult.Yes
Then

```

```

cmd = New SqlCommand("Delete from tblTeach
where IDNumber = '" &
grdAddProfessor.Item(3,
grdAddProfessor.CurrentRow.Index).
Value & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
End If
End If
End Sub

Private Sub btnAddSchedule_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnAddSchedule.Click
profid = txtID.Text.Trim
frmAddSchedule.Show()
End Sub

Private Sub btnRegisterPrint_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnRegisterPrint.Click
profid = txtID.Text.Trim
frmAddPrint.Show()
End Sub

Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnCancel.Click
Me.Close()
End Sub

Private Sub btnCancel2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnCancel2.Click
Clear()
cboDepartment.Show()
txtDepartment.Hide()
btnCancel2.Hide()
btnCancel2.Enabled = False
btnCancel.Show()
btnCancel.Enabled = True
btnDelete.Hide()
btnDelete.Enabled = False
btnAdd.Show()
btnAdd.Enabled = True
imgProfessor.ImageLocation = Nothing
End Sub

Private Sub btnBrowse_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnBrowse.Click
Dim BrowseImage As New OpenFileDialog
BrowseImage.Filter =
"jpeg|*.jpg|gif|*.gif|png|*.png|bitmap|*.bmp"
BrowseImage.ShowDialog()
imgProfessor.SizeMode =
PictureBoxSizeMode.StretchImage
imgProfessor.ImageLocation =
BrowseImage.FileName.ToString
End Sub
End Class

```

frmAddSchedule.vb

```

Imports System.Data.SqlClient
Public Class frmAddSchedule
Dim conn As SqlConnection
Private Sub frmAddSchedule_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
Dim test As New getconnstring

```

```

        conn = New SqlConnection(test.getconn)
        loadRoom()
        loadSubject()
        cboSubjectName.SelectedItem = 0
        cboRoom.SelectedItem = 0
        cboDay.SelectedItem = 0
        cboTimeIn.SelectedItem = 0
        cboTimeOut.SelectedItem = 0
        Dim ds As New DataSet
    da = New SqlDataAdapter("Select SubjectCode, Room,
        DayOfWeek, TimeIn, TimeOut From tblSchedule
        where IDNumber = '" & profid & "', conn)
        conn.Open()
        da.Fill(ds, "schedule")
        grdAddSchedule.DataSource =
ds.Tables("schedule")
        conn.Close()
    End Sub
    Private Sub loadRoom()
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select * from
tblRooms", conn)
        conn.Open()
        da.Fill(ds, "Rooms")
        cboRoom.DataSource = ds.Tables("Rooms")
        cboRoom.DisplayMember = "Room"
        conn.Close()
    End Sub
    Private Sub loadSubject()
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select * from
tblSubject", conn)
        conn.Open()
        da.Fill(ds, "subject")
        cboSubjectName.DataSource =
ds.Tables("subject")
        cboSubjectName.DisplayMember =
"SubjectName"
        conn.Close()
    End Sub
    Private Sub gridview()
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select SubjectCode, Room,
        DayOfWeek, TimeIn, TimeOut From
tblSchedule", conn)
        conn.Open()
        da.Fill(ds, "schedule")
        grdAddSchedule.DataSource =
ds.Tables("schedule")
        conn.Close()
    End Sub
    Private Sub
        cboSubjectName_SelectionChangeCommitted(By
        Val sender As System.Object, ByVal e As
        System.EventArgs) Handles
        cboSubjectName.SelectionChangeCommitted
        cmd = New SqlCommand("Select * from tblSubject
        where SubjectName = '" &
        cboSubjectName.Text.Trim & "'", conn)
        conn.Open()
        dr = cmd.ExecuteReader

        If dr.Read Then
            txtSubjectCode.Text = dr("SubjectCode")
            conn.Close()
        Exit Sub
        End If
        conn.Close()
    End Sub

Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnAdd.Click
    Dim Sked As String = Nothing
    Dim index As Char
    Dim start As Integer
    Dim last As Integer
    Dim ProfSked As String
    Dim rmsked As String
    Dim day As String
    Dim skedfinal As String = Nothing
    Dim rmskedfinal As String = Nothing
    Dim timelength As Integer
    day = cboDay.SelectedItem.ToString
    start = cboTimeIn.SelectedIndex
    last = cboTimeOut.SelectedIndex + 1
    If start > last - 1 Then
        MessageBox.Show("Invalid Time Inputs", "Error",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error)
    Exit Sub
    End If

    For i As Integer = 0 To 4
        If i >= start And i < last Then
            index = "1"
            Sked = Sked + index
        Else
            index = "0"
            Sked = Sked + index
        End If
    Next

    cmd = New SqlCommand("Select * from tblTeach
    where IDNumber = '" & profid & "', conn)
    conn.Open()
    dr = cmd.ExecuteReader

    If dr.Read Then
        ProfSked = dr(day)
        dr.Close()
    Else
        MessageBox.Show("Unknown Error", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error)
        dr.Close()
        conn.Close()
    Exit Sub
    End If
    conn.Close()

    For i As Integer = 0 To 4
        If Sked(i) = "1" And ProfSked(i) = "1" Then
            MessageBox.Show("Schedule is in conflict",
                "Error", MessageBoxButtons.OK,
                MessageBoxIcon.Error)
        Exit Sub
        ElseIf Sked(i) <> ProfSked(i) Then
            index = "1"
            skedfinal = skedfinal + index
        ElseIf Sked(i) = "0" And ProfSked(i) = "0"
        Then
            index = "0"
            skedfinal = skedfinal + index
        End If
    Next

    cmd = New SqlCommand("Select * from
tblRooms where Room = '" & cboRoom.Text & "'", conn)
    conn.Open()
    dr = cmd.ExecuteReader

```

```

If dr.Read Then
    rmsked = dr(day)
    dr.Close()
Else
    MessageBox.Show("Invalid Room Selection",
        "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error)
    dr.Close()
    conn.Close()
    Exit Sub
End If
conn.Close()

For i As Integer = 0 To 4
    If Sked(i) = "1" And rmsked(i) = "1" Then
        MessageBox.Show("Schedule is in conflict",
            "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error)
        Exit Sub
    ElseIf Sked(i) = "1" And rmsked(i) = "0" Then
        index = "1"
        rmskedfinal = rmskedfinal + index
        timelength = timelength + 5
    ElseIf Sked(i) = "0" And rmsked(i) = "0" Then
        index = "0"
        rmskedfinal = rmskedfinal + index
    ElseIf Sked(i) = "0" And rmsked(i) = "1" Then
        index = "1"
        rmskedfinal = rmskedfinal + index
    End If
Next

If day = "Monday" Then
    cmd = New SqlCommand("Update tblTeach Set
        Monday = " & skedfinal & " where
        IDNumber = " & profid & "", conn)
    conn.Open()
    cmd.ExecuteNonQuery()
    conn.Close()

    cmd = New SqlCommand("Update tblRooms Set
        Monday = " & rmskedfinal & " where
        Room = " & cboRoom.Text & "",
        conn)
    conn.Open()
    cmd.ExecuteNonQuery()
    conn.Close()

    ElseIf day = "Tuesday" Then
        cmd = New SqlCommand("Update tblTeach Set
            Tuesday = " & skedfinal & " where
            IDNumber = " & profid & "", conn)
        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()

        cmd = New SqlCommand("Update tblRooms Set
            Tuesday = " & rmskedfinal & " where
            Room = " & cboRoom.Text & "",
            conn)
        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()

        ElseIf day = "Wednesday" Then
            cmd = New SqlCommand("Update tblTeach Set
                Wednesday = " & skedfinal & "
                where IDNumber = " & profid & "",
                conn)
            conn.Open()
            cmd.ExecuteNonQuery()

            conn.Close()

            cmd = New SqlCommand("Update tblRooms Set
                Wednesday = " & rmskedfinal & "
                where Room = " & cboRoom.Text & "",
                conn)
            conn.Open()
            cmd.ExecuteNonQuery()
            conn.Close()

            cmd = New SqlCommand("Update tblRooms Set
                Wednesday = " & rmskedfinal & "
                where Room = " & cboRoom.Text &
                "", conn)
            conn.Open()
            cmd.ExecuteNonQuery()
            conn.Close()

            ElseIf day = "Thursday" Then
                cmd = New SqlCommand("Update tblTeach Set
                    Thursday = " & skedfinal & " where
                    IDNumber = " & profid & "", conn)
                conn.Open()
                cmd.ExecuteNonQuery()
                conn.Close()

                cmd = New SqlCommand("Update tblRooms Set
                    Thursday = " & rmskedfinal & "
                    where Room = " & cboRoom.Text &
                    "", conn)
                conn.Open()
                cmd.ExecuteNonQuery()
                conn.Close()

                ElseIf day = "Friday" Then
                    cmd = New SqlCommand("Update tblTeach Set
                        Friday = " & skedfinal & " where
                        IDNumber = " & profid & "", conn)
                    conn.Open()
                    cmd.ExecuteNonQuery()
                    conn.Close()

                    cmd = New SqlCommand("Update tblRooms Set
                        Friday = " & rmskedfinal & " where
                        Room = " & cboRoom.Text & "",
                        conn)
                    conn.Open()
                    cmd.ExecuteNonQuery()
                    conn.Close()

                    ElseIf day = "Saturday" Then
                        cmd = New SqlCommand("Update tblTeach Set
                            Saturday = " & skedfinal & " where
                            IDNumber = " & profid & "", conn)
                        conn.Open()
                        cmd.ExecuteNonQuery()
                        conn.Close()

                        cmd = New SqlCommand("Update tblRooms Set
                            Saturday = " & rmskedfinal & "
                            where Room = " & cboRoom.Text &
                            "", conn)
                        conn.Open()
                        cmd.ExecuteNonQuery()
                        conn.Close()

                        ElseIf day = "Sunday" Then
                            cmd = New SqlCommand("Update tblTeach Set
                                Sunday = " & skedfinal & " where
                                IDNumber = " & profid & "", conn)
                            conn.Open()
                            cmd.ExecuteNonQuery()
                            conn.Close()

                            cmd = New SqlCommand("Update tblRooms Set
                                Sunday = " & rmskedfinal & " where
                                Room = " & cboRoom.Text &
                                "", conn)
                            conn.Open()
                            cmd.ExecuteNonQuery()
                            conn.Close()

                            End If

```

```

dt = New DataTable
da = New SqlDataAdapter("Select * from
tblSchedule", conn)
conn.Open()
da.Fill(dt)
cmb = New SqlCommandBuilder(da)

newRow = dt.NewRow
newRow("IDNumber") = profid
newRow("SubjectName") =
cboSubjectName.Text.Trim
newRow("SubjectCode") =
txtSubjectCode.Text.Trim
newRow("Room") = cboRoom.Text.Trim
newRow("DayOfWeek") = cboDay.Text.Trim
newRow("TimeIn") = cboTimeIn.Text.Trim
newRow("TimeOut") = cboTimeOut.Text.Trim
newRow("BinSked") = Sked
newRow("TimeLength") = timelength
dt.Rows.Add(newRow)
da.InsertCommand = cmb.GetInsertCommand
da.Update(dt)
conn.Close()
gridview()
End Sub

Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnCancel.Click
Me.Close()
End Sub

Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
Dim BinSked As String = Nothing
Dim Room As String = Nothing
Dim DayOfWeek As String = Nothing
Dim RoomSked As String = Nothing
Dim NewRoomSked As String = Nothing
Dim ProfSked As String = Nothing
Dim NewProfSked As String = Nothing
Dim index As String

cmd = New SqlCommand("Select * from
tblSchedule where IDNumber = " &
& profid & " AND Room = " &
& grdAddSchedule.Item(1,
grdAddSchedule.CurrentRow.Index).Value & " AND
DayOfWeek = " &
& grdAddSchedule.Item(2,
grdAddSchedule.CurrentRow.Index).Value & " AND
TimeIn = " &
& grdAddSchedule.Item(3,
grdAddSchedule.CurrentRow.Index).Value & " AND
TimeOut = " &
& grdAddSchedule.Item(4,
grdAddSchedule.CurrentRow.Index).Value & "", conn)
conn.Open()
dr = cmd.ExecuteReader
If dr.Read Then
BinSked = dr("BinSked")
Room = dr("Room")
DayOfWeek = dr("DayOfWeek")
dr.Close()
conn.Close()
Else
dr.Close()
conn.Close()
End If

```

```

cmd = New SqlCommand("Select * from
tblRooms where Room = " & Room & "", conn)
conn.Open()
dr = cmd.ExecuteReader
If dr.Read Then
If DayOfWeek = "Monday" Then
RoomSked = dr("Monday")
ElseIf DayOfWeek = "Tuesday" Then
RoomSked = dr("Tuesday")
ElseIf DayOfWeek = "Wednesday" Then
RoomSked = dr("Wednesday")
ElseIf DayOfWeek = "Thursday" Then
RoomSked = dr("Thursday")
ElseIf DayOfWeek = "Friday" Then
RoomSked = dr("Friday")
ElseIf DayOfWeek = "Saturday" Then
RoomSked = dr("Saturday")
ElseIf DayOfWeek = "Sunday" Then
RoomSked = dr("Sunday")
End If
dr.Close()
conn.Close()
Else
dr.Close()
conn.Close()
End If
For i As Integer = 0 To 4
If RoomSked(i) = "1" And BinSked(i) = "1"
Then
index = "0"
NewRoomSked = NewRoomSked + index
Else
index = RoomSked(i)
NewRoomSked = NewRoomSked + index
End If
Next

If DayOfWeek = "Monday" Then
cmd = New SqlCommand("Update tblRooms Set
Monday = " & NewRoomSked & "
where Room = " & Room & "", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Tuesday" Then
cmd = New SqlCommand("Update tblRooms Set
Tuesday = " & NewRoomSked & "
where Room = " & Room & "", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Wednesday" Then
cmd = New SqlCommand("Update tblRooms Set
Wednesday = " & NewRoomSked & "
where Room = " & Room & "",
conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Thursday" Then
cmd = New SqlCommand("Update tblRooms Set
Thursday = " & NewRoomSked & "
where Room = " & Room & "", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Friday" Then

```

```

cmd = New SqlCommand("Update tblRooms Set
    Friday = " & NewRoomSked & "
    where Room = " & Room & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Saturday" Then
cmd = New SqlCommand("Update tblRooms Set
    Saturday = " & NewRoomSked & "
    where Room = " & Room & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Sunday" Then
cmd = New SqlCommand("Update tblRooms Set
    Sunday = " & NewRoomSked & "
    where Room = " & Room & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
End If

cmd = New SqlCommand("Select * from tblTeach
where IDNumber = " & profid & "'", conn)
conn.Open()
dr = cmd.ExecuteReader
If dr.Read Then
    If DayOfWeek = "Monday" Then
        ProfSked = dr("Monday")
    ElseIf DayOfWeek = "Tuesday" Then
        ProfSked = dr("Tuesday")
    ElseIf DayOfWeek = "Wednesday" Then
        ProfSked = dr("Wednesday")
    ElseIf DayOfWeek = "Thursday" Then
        ProfSked = dr("Thursday")
    ElseIf DayOfWeek = "Friday" Then
        ProfSked = dr("Friday")
    ElseIf DayOfWeek = "Saturday" Then
        ProfSked = dr("Saturday")
    ElseIf DayOfWeek = "Sunday" Then
        ProfSked = dr("Sunday")
    End If
    dr.Close()
    conn.Close()
Else
    dr.Close()
    conn.Close()
End If
For i As Integer = 0 To 4
    If ProfSked(i) = "1" And BinSked(i) = "1" Then
        index = "0"
        NewProfSked = NewProfSked + index
    Else
        index = ProfSked(i)
        NewProfSked = NewProfSked + index
    End If
Next

If DayOfWeek = "Monday" Then
cmd = New SqlCommand("Update tblTeach Set
    Monday = " & NewProfSked & "
    where IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Tuesday" Then
cmd = New SqlCommand("Update tblTeach Set
    Tuesday = " & NewProfSked & "
    where IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Wednesday" Then
cmd = New SqlCommand("Update tblTeach Set
    Wednesday = " & NewProfSked & "
    where IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Thursday" Then
cmd = New SqlCommand("Update tblTeach Set
    Thursday = " & NewProfSked & "
    where IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Friday" Then
cmd = New SqlCommand("Update tblTeach Set
    Friday = " & NewProfSked & " where
    IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Saturday" Then
cmd = New SqlCommand("Update tblTeach Set
    Saturday = " & NewProfSked & "
    where IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
ElseIf DayOfWeek = "Sunday" Then
cmd = New SqlCommand("Update tblTeach Set
    Sunday = " & NewProfSked & "
    where IDNumber = " & profid & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
End If

cmd = New SqlCommand("Delete from
tblSchedule where IDNumber = " _
    & profid & " AND Room = " _
    & grdAddSchedule.Item(1,
grdAddSchedule.CurrentRow.Index).Value & " AND
DayOfWeek = " _
    & grdAddSchedule.Item(2,
grdAddSchedule.CurrentRow.Index).Value & " AND
TimeIn = " _
    & grdAddSchedule.Item(3,
grdAddSchedule.CurrentRow.Index).Value & " AND
TimeOut = " _
    & grdAddSchedule.Item(4,
grdAddSchedule.CurrentRow.Index).Value & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
gridview()
End Sub
End Class

frmAdministrator.vb

Imports System.Data.SqlClient

```

```

Public Class frmAdministrator
    Dim conn As SqlConnection

    Private Sub frmAdministrator_Load(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim test As New getconnstring
        conn = New SqlConnection(test.getconn)
        btnDelete.Enabled = False
        btnCancel2.Enabled = False
        btnSave.Enabled = False
        btnSave.SendToBack()
        btnCancel2.Hide()
        btnCancel2.Enabled = False
        GridView()
    End Sub
    Private Sub GridView()
        Dim ds As New DataSet

        da = New SqlDataAdapter("Select
Surname,Firstname,Middlename,Username from
tblAdministrator", conn)
        conn.Open()
        da.Fill(ds, "administrator")
        grdAdministrator.DataSource =
ds.Tables("administrator")
        conn.Close()
        btnDelete.Enabled = False
        btnEdit.Enabled = False
    End Sub
    Private Sub Clear()
        txtSurname.Text = Nothing
        txtFirstname.Text = Nothing
        txtMiddlename.Text = Nothing
        txtUsername.Text = Nothing
        txtPassword.Text = Nothing
        txtConfirmNewPassword.Text = Nothing
        cboSecretQuestion.SelectedIndex = 0
        txtAnswer.Text = Nothing
    End Sub

    Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAdd.Click
        dt = New DataTable

        *****
        *****
        'Check for Information Deficiency
        *****
        *****
        If txtSurName.Text = Nothing Then
            MsgBox.Show("The Surname field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
        ElseIf txtFirstName.Text = Nothing Then
            MsgBox.Show("The First Name field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
        ElseIf txtUserName.Text = Nothing Then
            MsgBox.Show("The Username field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
        ElseIf txtPassword.Text = Nothing Then
            MsgBox.Show("The Password field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)

            Exit Sub
        ElseIf txtConfirmNewPassword.Text = Nothing
Then
            MsgBox.Show("Please Confirm your
password", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
            Exit Sub
        ElseIf
cboSecretQuestion.Items(cboSecretQuestion.SelectedIndex)
= Nothing Then
            MsgBox.Show("You did not select a secret
question", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
            Exit Sub
        ElseIf txtAnswer.Text = Nothing Then
            MsgBox.Show("The answer field is empty",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If
        *****
        *****
        Dim wrap_password As New
encrypt.Simple3Des(txtPassword.Text.Trim)
        Dim wrap_confirm As New
encrypt.Simple3Des(txtConfirmNewPassword.Text.Trim)
        Dim wrap_answer As New
encrypt.Simple3Des(txtAnswer.Text.Trim)
        Dim ciphertext_password As String =
wrap_password.EncryptData(txtPassword.Text.Trim)
        Dim ciphertext_confirm As String =
wrap_confirm.EncryptData(txtConfirmNewPassword.Text.Tr
im)
        Dim ciphertext_answer As String =
wrap_answer.EncryptData(txtAnswer.Text.Trim)

        If ciphertext_password <> ciphertext_confirm
Then
            MsgBox.Show("Please retype password",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If
        da = New SqlDataAdapter("Select * from
tblAdministrator", conn)
        conn.Open()
        da.Fill(dt)
        cmb = New SqlCommandBuilder(da)
        *****
        *****
        'Check for Redundancy
        *****
        *****
        cmd = New SqlCommand("Select Username from
tblAdministrator where Username = " &
txtUserName.Text.Trim & """, conn)
        dr = cmd.ExecuteReader
        If dr.Read Then
            MsgBox.Show("Username already exists in
the database", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            dr.Close()
            conn.Close()
            dr.Close()
            Exit Sub
        Else
            dr.Close()
        End If
        *****
        *****

```

```

*****
*****
newRow = dt.NewRow
newRow("Surname") = txtSurName.Text.Trim
newRow("Firstname") = txtFirstName.Text.Trim
newRow("Middlename") =
txtMiddleName.Text.Trim
newRow("Username") = txtUserName.Text.Trim
newRow("Password") = ciphertext_password
newRow("SecretQuestion") =
cboSecretQuestion.Items(cboSecretQuestion.SelectedIndex)
newRow("Answer") = ciphertext_answer
dt.Rows.Add(newRow)

da.InsertCommand = cmb.GetInsertCommand
da.Update(dt)
conn.Close()
GridView()
Clear()
End Sub

Private Sub btnEdit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnEdit.Click
    cmd = New SqlCommand("Select * from
tblAdministrator where Username = " & adminuser & """,
conn)

    conn.Open()
    dr = cmd.ExecuteReader

    If dr.Read Then
        txtSurName.Text = dr("Surname")
        txtFirstName.Text = dr("Firstname")
        txtMiddleName.Text = dr("Middlename")
        txtUserName.Text = dr("Username")
        txtUserName.Enabled = False
        txtSurName.Enabled = False
        txtFirstName.Enabled = False
        txtMiddleName.Enabled = False
        cboSecretQuestion.Enabled = False
        txtAnswer.Enabled = False
        btnEdit.SendToBack()
        btnEdit.Enabled = False
        btnSave.BringToFront()
        btnSave.Enabled = True
        btnCancel.Enabled = False
        btnCancel2.BringToFront()
        btnCancel2.Enabled = True
        dr.Close()
        conn.Close()
    Else
        dr.Close()
        conn.Close()
    End If
End Sub

Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnSave.Click
    Dim wrap_newpass As New
encrypt.Simple3Des(txtPassword.Text.Trim)
    Dim ciphertext_newpass As String =
wrap_newpass.EncryptData(txtPassword.Text.Trim)
    Dim wrap_passconfirm As New
encrypt.Simple3Des(txtConfirmNewPassword.Text.Trim)
    Dim ciphertext_passconfirm As String =
wrap_newpass.EncryptData(txtConfirmNewPassword.Text.
Trim)

```

```

If ciphertext_newpass = ciphertext_passconfirm
Then
    cmd = New SqlCommand("Update
tblAdministrator Set Password = " & ciphertext_newpass &
" where Username = " & txtUserName.Text.Trim & """,
conn)

    conn.Open()
    cmd.ExecuteNonQuery()
    conn.Close()
Else
    MessageBox.Show("Please retype password",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    Exit Sub
End If
btnSave.SendToBack()
btnSave.Enabled = False
btnCancel2.SendToBack()
btnCancel2.Enabled = False
Clear()
GridView()
End Sub

Private Sub grdAdministrator_CellClick(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs)
Handles grdAdministrator.CellClick
    btnDelete.Enabled = True
    btnEdit.Enabled = True
    btnCancel.Enabled = False
    btnCancel.Hide()
    btnCancel2.Show()
    btnCancel2.Enabled = True
End Sub

Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnDelete.Click
    Dim response As DialogResult

    response = MessageBox.Show("Are you sure you
want to delete the selected item?", "Warning",
MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation)
    If response = Windows.Forms.DialogResult.Yes
Then
        conn.Open()
        cmd = New SqlCommand("Delete from
tblAdministrator where Username = " &
grdAdministrator.Item(3,
grdAdministrator.CurrentRow.Index).Value & """, conn)
        cmd.ExecuteNonQuery()
        conn.Close()
        GridView()
        ElseIf response =
Windows.Forms.DialogResult.No Then
            GridView()
            btnDelete.Enabled = False
            Exit Sub
        End If
    End Sub

Private Sub btnOk_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnOk.Click
    Me.Close()
End Sub

Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
    Me.Close()

```

```

End Sub

Private Sub btnCancel2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel2.Click
    Clear()
    Gridview()
    txtSurName.Enabled = True
    txtFirstName.Enabled = True
    txtMiddleName.Enabled = True
    txtPassword.Enabled = True
    txtConfirmNewPassword.Enabled = True
    txtAnswer.Enabled = True
    cboSecretQuestion.Enabled = True
    btnSave.SendToBack()
    btnCancel2.SendToBack()
    btnSave.Enabled = False
    btnCancel2.Enabled = False
    btnEdit.Enabled = False
    btnCancel.Show()
    btnCancel.Enabled = True
    btnCancel2.Enabled = False
    btnCancel2.Hide()
    btnDelete.Enabled = False
End Sub
End Class

frmExecute.vb

Imports GrFingerXLib
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient

Public Class frmExecute
    Inherits System.Windows.Forms.Form
    Dim myUtil As Util
    Dim unlock As Unlock

    Private Sub frmExecute_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim err As Integer
        ' initialize util class
        myUtil = New Util(LogList, PictureBox1,
AxGrFingerXCtrl1)
        unlock = New Unlock
        ' Initialize GrFingerX Library
        err = myUtil.InitializeGrFinger()
        ' Print result in log
        If err < 0 Then
            myUtil.WriteError(err)
            Exit Sub
        Else
            myUtil.WriteLog("**GrFingerX Initialized
Successful**")
        End If
        frmMain.btnExecute.Enabled = False
    End Sub

    Private Sub frmExecute_FormClosed(ByVal sender
As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles
Me.FormClosed
        myUtil.FinalizeGrFinger()
        frmMain.btnExecute.Enabled = True
    End Sub

    ' A fingerprint reader was plugged on system
    Private Sub AxGrFingerXCtrl1_SensorPlug(ByVal
sender As System.Object, ByVal e As

```

```

AxGrFingerXLib._IGrFingerXCtrlEvents_SensorPlugEvent)
Handles AxGrFingerXCtrl1.SensorPlug
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Plugged.")
        AxGrFingerXCtrl1.CapStartCapture(e.idSensor)
    End Sub

    ' A fingerprint reader was unplugged from system
    Private Sub
AxGrFingerXCtrl1_SensorUnplug(ByVal sender As
System.Object, ByVal e As
AxGrFingerXLib._IGrFingerXCtrlEvents_SensorUnplugEve
nt) Handles AxGrFingerXCtrl1.SensorUnplug
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Unplugged.")
        AxGrFingerXCtrl1.CapStopCapture(e.idSensor)
    End Sub

    ' A finger was placed on reader
    Private Sub AxGrFingerXCtrl1_FingerDown(ByVal
sender As System.Object, ByVal e As
AxGrFingerXLib._IGrFingerXCtrlEvents_FingerDownEvent
) Handles AxGrFingerXCtrl1.FingerDown
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Finger Placed.")
    End Sub

    ' An image was acquired from reader
    Private Sub
AxGrFingerXCtrl1_ImageAcquired(ByVal sender As
System.Object, ByVal e As
AxGrFingerXLib._IGrFingerXCtrlEvents_ImageAcquiredEv
ent) Handles AxGrFingerXCtrl1.ImageAcquired
        ' Copying aquired image
        myUtil.raw.height = e.height
        myUtil.raw.width = e.width
        myUtil.raw.res = e.res
        myUtil.raw.img = e.rawImage

        ' Signaling that an Image Event occurred.
        myUtil.WriteLog("Sensor: " & e.idSensor & ".
Event: Image captured.")

        ' display fingerprint image
        myUtil.PrintBiometricDisplay(False,
GRConstants.GR_DEFAULT_CONTEXT)
        ' now we have a fingerprint, so we can extract
        template
        *****
        "Template Extraction
        *****
        Dim ret As Integer
        ' extract template
        ret = myUtil.ExtractTemplate()
        ' write template quality to log
        If ret = GRConstants.GR_BAD_QUALITY Then
            myUtil.WriteLog("Template extracted
successfully. Bad quality.")
        ElseIf ret =
GRConstants.GR_MEDIUM_QUALITY Then
            myUtil.WriteLog("Template extracted
successfully. Medium quality.")
        ElseIf ret = GRConstants.GR_HIGH_QUALITY
Then
            myUtil.WriteLog("Template extracted
successfully. High quality.")
        End If
        If ret >= 0 Then

```

```

        ' if no error, display
        minutiae/segments/directions into the image
        myUtil.PrintBiometricDisplay(True,
GRConstants.GR_NO_CONTEXT)
        ' enable operations we can do over extracted
        template
        Else
            ' write error to log
            myUtil.WriteError(ret)
        End If
        *****
        *****
        *****
        Dim score As Integer
        score = 0
        ' identify it
        ret = myUtil.Identify(score)
        ' write result to log
        If ret > 0 Then
            myUtil.WriteLog("Fingerprint identified. ID = "
& ret & ". Score = " & score & ".")
            myUtil.PrintBiometricDisplay(True,
GRConstants.GR_DEFAULT_CONTEXT)
        ElseIf ret = 0 Then
            myUtil.WriteLog("Fingerprint not Found.")
            Exit Sub
        Else
            myUtil.WriteError(ret)
            Exit Sub
        End If
        inputid = ret
        sensorid = e.idSensor
        unlock.DoorRelease()
    End Sub
    Private Sub TimerOpenClose_Tick(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TimerOpenClose.Tick
        frmMain.MSComm.Open()
        frmMain.MSComm.Write(127)
        frmMain.MSComm.Close()
        TimerOpenClose.Stop()
    End Sub
End Class

```

frmForgotPassword.vb

```
Imports System.Data.SqlClient
```

```
Public Class frmForgotPassword
```

```
    Dim conn As SqlConnection
```

```
    Dim tempAnswer As String
```

```
    Private Sub frmForgotPassword_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
```

```
        txtAnswer.Enabled = False
```

```
        btnVerify.Enabled = False
```

```
        Dim test As New getconnstring
```

```
        conn = New SqlConnection(test.getconn)
```

```
    End Sub
```

```
    Private Sub btnRetrievePassword_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRetrievePassword.Click
```

```
        If txtUsername.Text = Nothing Then
```

```
            MessageBox.Show("Username field is empty",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
```

```
            Exit Sub
```

```
        End If
```

```
        cmd = New SqlCommand("Select * from
tblAdministrator where Username=" &
txtUsername.Text.Trim & "", conn)
        conn.Open()
        dr = cmd.ExecuteReader
```

```
    If dr.Read Then
```

```
        lblSecretQuestion.Text = dr("SecretQuestion")
```

```
        txtAnswer.Enabled = True
```

```
        tempAnswer = dr("Answer")
```

```
        btnVerify.Enabled = True
```

```
        btnRetrievePassword.Enabled = False
```

```
        dr.Close()
```

```
        conn.Close()
```

```
    Else
```

```
        MessageBox.Show("Username does not exist",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
```

```
        conn.Close()
```

```
        Exit Sub
```

```
    End If
```

```
End Sub
```

```
Private Sub btnVerify_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnVerify.Click
```

```
    Dim wrap_answer As New
```

```
encrypt.Simple3Des(txtAnswer.Text.Trim)
```

```
    Dim ciphertext_answer As String =
```

```
wrap_answer.EncryptData(txtAnswer.Text.Trim)
```

```
    If ciphertext_answer = tempAnswer Then
```

```
        btnVerify.Enabled = False
```

```
        Randomize()
```

```
        Dim ticket As String = ""
```

```
        Dim ctr As Int16 = 5
```

```
        While ctr <> 0
```

```
            ticket = ticket + (CStr(CInt(Rnd() * 9)))
```

```
            ctr = ctr - 1
```

```
        End While
```

```
        lblPassword.Text = ticket
```

```
        Dim wrap_newpass As New
```

```
encrypt.Simple3Des(lblPassword.Text.Trim)
```

```
        Dim ciphertext_newpass As String =
```

```
wrap_newpass.EncryptData(lblPassword.Text.Trim)
```

```
        cmd = New SqlCommand("UPDATE
tblAdministrator SET Password=" & ciphertext_newpass &
" WHERE Username=" & txtUsername.Text.Trim & "",
conn)
```

```
        conn.Open()
```

```
        cmd.ExecuteNonQuery()
```

```
        conn.Close()
```

```
    Else
```

```
        MessageBox.Show("Your Answer is incorrect",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
```

```
        Exit Sub
```

```
    End If
```

```
End Sub
```

```
Private Sub btnOk_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnOk.Click
    Me.Close()
End Sub
```

```
Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
```

```

        Me.Close()
    End Sub
End Class

frmItems.vb

Imports System.Data.SqlClient
Public Class frmItems
    Dim conn As SqlConnection
    Private Sub frmItems_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim test As New getconnstring
        conn = New SqlConnection(test.getconn)
        btnDelete.Enabled = False
    End Sub
    Private Sub TabRooms_Enter(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
TabRooms.Enter
        btnDelete.Enabled = False
        Dim ds As New DataSet

        da = New SqlDataAdapter("Select Room,
RoomSensor From tblRooms", conn)
        conn.Open()
        da.Fill(ds, "rooms")
        grdItems.DataSource = ds.Tables("rooms")
        conn.Close()
    End Sub
    Private Sub TabSubject_Enter(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
TabSubject.Enter
        btnDelete.Enabled = False
        Dim ds As New DataSet

        da = New SqlDataAdapter("Select * From
tblSubject", conn)
        conn.Open()
        da.Fill(ds, "subject")
        grdItems.DataSource = ds.Tables("subject")
        conn.Close()
    End Sub

    Private Sub TabDepartment_Enter(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
TabDepartment.Enter
        btnDelete.Enabled = False
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select * From
tblDepartment", conn)
        conn.Open()
        da.Fill(ds, "department")
        grdItems.DataSource = ds.Tables("department")
        conn.Close()
    End Sub

    Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAdd.Click
        dt = New DataTable
        If TabItems.SelectedTab.Name = "TabRooms"
Then
            If txtRoom.Text = Nothing Then
                MessageBox.Show("The Room Field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
            ElseIf txtRoomSensor.Text = Nothing Then

```

```

                MessageBox.Show("The Sensor Number
Field is empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
            ElseIf txtLockAddress.Text = Nothing Then
                MessageBox.Show("The Lock Address Field
is empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
        End If
        da = New SqlDataAdapter("SELECT * FROM
tblRooms", conn)
        conn.Open()
        da.Fill(dt)
        cmb = New SqlCommandBuilder(da)
        *****
        'Check for Room Redundancy
        *****
        cmd = New SqlCommand("Select RoomId
From tblRooms WHERE Room=" & txtRoom.Text.Trim &
"", conn)
        dr = cmd.ExecuteReader
        If dr.Read Then
            MessageBox.Show("The Room already exists
in the database", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            dr.Close()
            conn.Close()
            Exit Sub
        Else
            dr.Close()
        End If
        *****
        *****
        newRow = dt.NewRow
        newRow("Room") = txtRoom.Text.Trim
        newRow("RoomSensor") =
txtRoomSensor.Text.Trim
        newRow("RoomID") = DBNull.Value
        newRow("Monday") = "00000"
        newRow("Tuesday") = "00000"
        newRow("Wednesday") = "00000"
        newRow("Thursday") = "00000"
        newRow("Friday") = "00000"
        newRow("Saturday") = "00000"
        newRow("Sunday") = "00000"
        newRow("LockAddress") =
txtLockAddress.Text.Trim
        newRow("LockStatus") = "0"
        dt.Rows.Add(newRow)

        da.InsertCommand = cmb.GetInsertCommand
        da.Update(dt)
        conn.Close()
        TabRooms_Enter(sender, e)
        ElseIf TabItems.SelectedTab.Name =
"TabSubject" Then
            If txtSubjectName.Text = Nothing Then
                MessageBox.Show("Subject Name field is
empty", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Exit Sub
            ElseIf txtSubjectCode.Text = Nothing Then
                MessageBox.Show("Subject Code field is
empty", "error", MessageBoxButtons.OK,
MessageBoxIcon.Error)

```

```

        Exit Sub
    End If
    da = New SqlDataAdapter("SELECT * FROM
tblSubject", conn)
    conn.Open()
    da.Fill(dt)
    cmb = New SqlCommandBuilder(da)
    *****
    'Check for Subject Redundancy
    *****
    cmd = New SqlCommand("Select SubjectCode
From tblSubject WHERE SubjectCode=" &
txtSubjectCode.Text.Trim & "'", conn)
    dr = cmd.ExecuteReader
    If dr.Read Then
        MsgBox.Show("The Subject Code
already exists in the database", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        dr.Close()
        conn.Close()
        Exit Sub
    Else
        dr.Close()
    End If
    *****
    *****
    *****
    newRow = dt.NewRow
    newRow("SubjectName") =
txtSubjectName.Text.Trim
    newRow("SubjectCode") =
txtSubjectCode.Text.Trim
    dt.Rows.Add(newRow)
    da.InsertCommand = cmb.GetInsertCommand
    da.Update(dt)
    conn.Close()
    TabSubject_Enter(sender, e)
    ElseIf TabItems.SelectedTab.Name =
"TabDepartment" Then
        If txtDepartmentName.Text = Nothing Then
            MsgBox.Show("Department Name Field
is empty", "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error)
            Exit Sub
        ElseIf txtDepartmentCode.Text = Nothing Then
            MsgBox.Show("Department Code Field
is empty", "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error)
            Exit Sub
        End If
        da = New SqlDataAdapter("Select * From
tblDepartment", conn)
        conn.Open()
        da.Fill(dt)
        cmb = New SqlCommandBuilder(da)
        *****
        'Check for Department Redundancy
        *****
        cmd = New SqlCommand("Select
DepartmentCode from tblDepartment where
DepartmentCode = " & txtDepartmentCode.Text.Trim & "'",
conn)
        dr = cmd.ExecuteReader
        If dr.Read Then

```

```

        MsgBox.Show("The Department Code
already exist in the database", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        dr.Close()
        conn.Close()
        Exit Sub
    Else
        dr.Close()
    End If
    *****
    *****
    *****
    newRow = dt.NewRow
    newRow("DepartmentName") =
txtDepartmentName.Text.Trim
    newRow("DepartmentCode") =
txtDepartmentCode.Text.Trim
    dt.Rows.Add(newRow)
    da.InsertCommand = cmb.GetInsertCommand
    da.Update(dt)
    conn.Close()
    TabDepartment_Enter(sender, e)
    End If
End Sub

Private Sub grdItems_CellClick(ByVal sender As
Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs)
Handles grdItems.CellClick
    btnDelete.Enabled = True
End Sub

Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnDelete.Click
    Dim response As DialogResult

    response = MsgBox.Show("Are you sure you
want to delete the selected row?", "Warning",
    MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation)
    If response = Windows.Forms.DialogResult.Yes
Then
        conn.Open()
        If TabItems.SelectedTab.Name = "TabRoom"
Then
            cmd = New SqlCommand("Delete from
tblRooms where Room = " & grdItems.Item(0,
grdItems.CurrentRow.Index).Value & "'", conn)
            cmd.ExecuteNonQuery()
            conn.Close()
            TabRooms_Enter(sender, e)
        ElseIf TabItems.SelectedTab.Name =
"TabSubject" Then
            cmd = New SqlCommand("Delete from
tblSubject where SubjectCode = " & grdItems.Item(1,
grdItems.CurrentRow.Index).Value & "'", conn)
            cmd.ExecuteNonQuery()
            conn.Close()
            TabSubject_Enter(sender, e)
        ElseIf TabItems.SelectedTab.Name =
"TabDepartment" Then
            cmd = New SqlCommand("Delete from
tblDepartment where DepartmentName = " &
grdItems.Item(0, grdItems.CurrentRow.Index).Value & "'",
conn)
            cmd.ExecuteNonQuery()
            conn.Close()
            TabDepartment_Enter(sender, e)
        Else

```

```

        conn.Close()
    End If
    ElseIf response =
Windows.Forms.DialogResult.No Then
        Exit Sub
    End If
End Sub
Private Sub btnOk_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnOk.Click
    Me.Close()
End Sub
Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
    Me.Close()
End Sub
End Class

```

frmLogIn.vb

```

Imports System.Data.SqlClient

Public Structure DoorClosedList
    Public Room As String
    Public LockAddress As String
    Public TimeOut As String
End Structure

Public Class frmMain
    Dim conn As SqlConnection

    Private Sub frmMain_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim test As New getconnstring
        conn = New SqlConnection(test.getconn)
        btnItems.Enabled = False
        Me.btnItems.Hide()
        Me.btnProfessor.Hide()
        Me.btnLogOut.Hide()
        Me.btnLogIn.Hide()
        Me.btnAdministrator.Hide()
        Me.btnCancel.Hide()
        Me.btnRoomAccess.Hide()
        btnProfessor.Enabled = False
        btnAdministrator.Enabled = False
        btnLogOut.SendToBack()
        tmrAutoClose.Start()
    End Sub

    Private Sub btnLogIn_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnLogIn.Click
        frmLogIn.Show()
    End Sub

    Private Sub btnLogOut_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnLogOut.Click
        adminuser = Nothing
        btnItems.Enabled = False
        btnProfessor.Enabled = False
        btnAdministrator.Enabled = False
        Me.btnCancel.Hide()
        Me.btnItems.Hide()
        Me.btnProfessor.Hide()
        Me.btnAdministrator.Hide()
        Me.btnLogIn.Hide()
        Me.btnLogOut.Hide()
    End Sub

```

```

        Me.btnExit.Show()
        Me.btnAdminControl.Show()
        Me.btnExecute.Show()
        Me.btnRoomAccess.Hide()
        btnLogIn.BringToFront()
    End Sub

    Private Sub btnItems_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnItems.Click
        frmItems.Show()
    End Sub

    Private Sub btnProfessor_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnProfessor.Click
        frmAddProfessor.Show()
    End Sub

```

```

    Private Sub btnExit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnExit.Click
        Me.Close()
    End Sub

    Private Sub btnExecute_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnExecute.Click
        frmExecute.Show()
    End Sub

```

```

    Private Sub btnAdministrator_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAdministrator.Click
        frmAdministrator.Show()
    End Sub

```

```

    Private Sub tmrAutoClose_Tick(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
tmrAutoClose.Tick
        Dim SystemTime As String =
Date.Now.ToShortTimeString
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select * from tblDoorLock
where TimeOut = '" & SystemTime & "'", conn)
        conn.Open()
        da.Fill(ds)
        Dim DCList As DoorClosedList()
        Dim list As DataRowCollection = ds.Tables(0).Rows
        ReDim DCList(list.Count)

```

```

        If list.Count = 0 Then
            conn.Close()
            Exit Sub
        End If
        For i As Integer = 1 To list.Count
            DCList(i).Room = list.Item(i - 1).Item("Room")
            DCList(i).LockAddress = list.Item(i -
1).Item("LockAddress")
            DCList(i).TimeOut = list.Item(i - 1).Item("TimeOut")
        Next
        conn.Close()

```

```

        For i As Integer = 1 To DCList.Length
            If i < DCList.Length Then
                Dim Room As String = DCList(i).Room
                cmd = New SqlCommand("Update tblRooms Set
LockStatus = 0 where Room = '" & Room & "'", conn)
                conn.Open()
                cmd.ExecuteNonQuery()
            End If
        Next
    End Sub

```

```

conn.Close()

'SEND SIGNAL PULSE TO CLOSE DOOR
MSComm.Open()
MSComm.Write(127)
MSComm.Close()

cmd = New SqlCommand("Delete from
tblDoorLock where Room = " & Room & "'", conn)
conn.Open()
cmd.ExecuteNonQuery()
conn.Close()
'warning
frmwarning.Show()
Else
Exit Sub
End If
Next
End Sub

Private Sub btnAdminControl_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAdminControl.Click
Me.btnLogIN.Show()
Me.btnExit.Hide()
Me.btnCancel.Show()
End Sub

Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
Me.btnLogIN.Hide()
Me.btnCancel.Hide()
Me.btnExit.Show()
End Sub

Private Sub btnRoomAccess_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRoomAccess.Click
frmRoomoAccess.Show()
End Sub
End Class

frmRoomAccess.vb

Imports System.Data.SqlClient
Public Class frmRoomoAccess
Private Sub btnBack_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBack.Click
Me.Close()
End Sub

Private Sub btnSearch_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnSearch.Click
If cboRoom.Text = Nothing Then
MessageBox.Show("No Room is Selected",
"Required Field", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
cboRoom.Focus()
End If

Dim getConn As New getconnstring
Dim conn As SqlConnection
conn = New SqlConnection(getConn.getconn)

```

```

Dim ds As New DataSet
Dim da As SqlDataAdapter

Try
da = New SqlDataAdapter("SELECT Name, IDNum,
Timein, Timeout FROM tblRoomAccess Where Room = " &
cboRoom.Text.Trim & " AND Recdate = " &
dtDateAccess.Text.Trim & "'", conn)
conn.Open()
da.Fill(ds, "Access")
If ds.Tables("Access").Rows.Count = 0 Then
MessageBox.Show("No Match Found", "Search
Result", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
dtDateAccess.Focus()
DataGridView1.DataSource = Nothing
Else
DataGridView1.DataSource = ds.Tables("Access")
DataGridView1.Columns(0).HeaderText = "Name"
DataGridView1.Columns(1).HeaderText = "ID
Number"
DataGridView1.Columns(2).HeaderText = "Time
In"
DataGridView1.Columns(3).HeaderText = "Time
Out"
End If
Catch ex As Exception
MessageBox.Show(ex.ToString, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
Finally
conn.Close()
End Try

End Sub

Private Sub frmRoomoAccess_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
Dim conn As SqlConnection
Dim getConn As New getconnstring
Dim da As SqlDataAdapter
Dim ds As New DataSet

conn = New SqlConnection(getConn.getconn)

da = New SqlDataAdapter("SELECT Room FROM
tblRooms", conn)
conn.Open()
da.Fill(ds, "ROM")
If ds.Tables("ROM").Rows.Count = 0 Then
MessageBox.Show("No rooms are available", "",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
Else
cboRoom.DataSource = ds.Tables("ROM")
cboRoom.DisplayMember = "Room"
End If
conn.Close()
End Sub
End Class

frmMain.vb

Imports System.Data.SqlClient

Public Structure DoorClosedList
Public Room As String
Public LockAddress As String
Public TimeOut As String
End Structure

```

```

Public Class frmMain
    Dim conn As SqlConnection

    Private Sub frmMain_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim test As New getconnstring
        conn = New SqlConnection(test.getconn)
        btnItems.Enabled = False
        Me.btnItems.Hide()
        Me.btnProfessor.Hide()
        Me.btnLogOut.Hide()
        Me.btnLogIN.Hide()
        Me.btnAdministrator.Hide()
        Me.btnCancel.Hide()
        Me.btnRoomAccess.Hide()
        btnProfessor.Enabled = False
        btnAdministrator.Enabled = False
        btnLogOut.SendToBack()
        tmrAutoClose.Start()
    End Sub

    Private Sub btnLogIN_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnLogIN.Click
        frmLogIn.Show()
    End Sub

    Private Sub btnLogOut_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnLogOut.Click
        adminuser = Nothing
        btnItems.Enabled = False
        btnProfessor.Enabled = False
        btnAdministrator.Enabled = False
        Me.btnCancel.Hide()
        Me.btnItems.Hide()
        Me.btnProfessor.Hide()
        Me.btnAdministrator.Hide()
        Me.btnLogIN.Hide()
        Me.btnLogOut.Hide()
        Me.btnExit.Show()
        Me.btnAdminControl.Show()
        Me.btnExecute.Show()
        Me.btnRoomAccess.Hide()
        btnLogIN.BringToFront()
    End Sub

    Private Sub btnItems_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnItems.Click
        frmItems.Show()
    End Sub

    Private Sub btnProfessor_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnProfessor.Click
        frmAddProfessor.Show()
    End Sub

    Private Sub btnExit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnExit.Click
        Me.Close()
    End Sub

    Private Sub btnExecute_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnExecute.Click
        frmExecute.Show()

```

```

End Sub

    Private Sub btnAdministrator_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAdministrator.Click
        frmAdministrator.Show()
    End Sub

    Private Sub tmrAutoClose_Tick(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
tmrAutoClose.Tick
        Dim SystemTime As String =
Date.Now.ToShortTimeString
        Dim ds As New DataSet
        da = New SqlDataAdapter("Select * from tblDoorLock
where TimeOut = '" & SystemTime & "'", conn)
        conn.Open()
        da.Fill(ds)
        Dim DCList As DoorClosedList()
        Dim list As DataRowCollection = ds.Tables(0).Rows
        ReDim DCList(list.Count)

        If list.Count = 0 Then
            conn.Close()
            Exit Sub
        End If
        For i As Integer = 1 To list.Count
            DCList(i).Room = list.Item(i - 1).Item("Room")
            DCList(i).LockAddress = list.Item(i -
1).Item("LockAddress")
            DCList(i).TimeOut = list.Item(i - 1).Item("TimeOut")
        Next
        conn.Close()

        For i As Integer = 1 To DCList.Length
            If i < DCList.Length Then
                Dim Room As String = DCList(i).Room
                cmd = New SqlCommand("Update tblRooms Set
LockStatus = 0 where Room = '" & Room & "'", conn)
                conn.Open()
                cmd.ExecuteNonQuery()
                conn.Close()

                'SEND SIGNAL PULSE TO CLOSE DOOR
                MSComm.Open()
                MSComm.Write(127)
                MSComm.Close()

                cmd = New SqlCommand("Delete from
tblDoorLock where Room = '" & Room & "'", conn)
                conn.Open()
                cmd.ExecuteNonQuery()
                conn.Close()
                'warning
                frmwarning.Show()
            Else
                Exit Sub
            End If
        Next
    End Sub

    Private Sub btnAdminControl_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAdminControl.Click
        Me.btnLogIN.Show()
        Me.btnExit.Hide()
        Me.btnCancel.Show()
    End Sub

```

```

Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
    Me.btnLogIn.Hide()
    Me.btnCancel.Hide()
    Me.btnExit.Show()
End Sub

```

```

Private Sub btnRoomAccess_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRoomAccess.Click
    frmRoomoAccess.Show()
End Sub

```

End Class

frmWarning.vb

```

Public Class frmwarning
    Dim ulock As Unlock
    Dim count As Integer

```

```

Private Sub frmwarning_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ulock = New Unlock
    mplayer.Ctlcontrols.play()
End Sub

```

```

Private Sub btnOk_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnOk.Click
    frmMain.MSComm.Open()
    frmMain.MSComm.Write(127)
    frmMain.MSComm.Close()
    frmExecute.TimerOpenClose.Start()
    mplayer.Ctlcontrols.stop()
    frmExecute.txtTimeOut.Text =
Date.Now.ToShortTimeString
    ulock.accessrecord()
    Me.Close()
End Sub

```

```

Private Sub btnCancel_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCancel.Click
    frmExecute.txtTimeOut.Text =
Date.Now.ToShortTimeString
    mplayer.Ctlcontrols.stop()
    ulock.accessrecord()
    Me.Close()
End Sub

```

End Class

Unlock.vb

```

Imports System.Data.SqlClient
Imports Microsoft.VisualBasic

```

```

Public Structure SchedList
    Public TimeIn As String
    Public TimeOut As String
    Public BinSked As String
    Public TimeLength As Integer
End Structure

```

```

Public Structure TimeList
    Public TimeIn As String
    Public TimeOut As String

```

End Structure

```

Public Class Unlock
    Dim conn As SqlConnection
    Dim tbTimeIn As String

```

```

Public Sub DoorRelease()
    Dim test As New getconnstring
    conn = New SqlConnection(test.getconn)

```

```

    Dim day As String = Date.Today.DayOfWeek.ToString
    Dim identifiedid As String
    Dim personaid As String
    Dim IdentifiedRoom As String = Nothing
    Dim LockAddress As String = Nothing
    Dim tbTimeOut As String
    Dim tbTimelength As Integer
    Dim TList As TimeList()
    ReDim TList(5)

```

```

    TList(0).TimeIn = "7:00 AM"
    TList(0).TimeOut = "7:05 AM"
    TList(1).TimeIn = "7:05 AM"
    TList(1).TimeOut = "7:10 AM"
    TList(2).TimeIn = "7:10 AM"
    TList(2).TimeOut = "7:15 AM"
    TList(3).TimeIn = "7:15 AM"
    TList(3).TimeOut = "7:20 AM"
    TList(4).TimeIn = "7:20 AM"
    TList(4).TimeOut = "7:25 AM"
    identifiedid = inputid
    personaid = sensorid

```

```

*****
*****
*****
'Search Door to be Unlocked

```

```

*****
*****
*****
cmd = New SqlCommand("Select * from tblRooms
where RoomSensor = '' & personaid & ''", conn)
conn.Open()
dr = cmd.ExecuteReader

```

```

If dr.Read Then
    IdentifiedRoom = dr("Room")
    LockAddress = dr("LockAddress")
    dr.Close()
    conn.Close()
Else
    dr.Close()
    conn.Close()
End If

```

```

*****
*****
*****

```

```

*****
*****
*****

```

```

*****
*****
*****
'Get System Time Binary Schedule

```

```

*****
*****
*****
Dim SystemTime As String =
Date.Now.ToShortTimeString
Dim SystemSked As String = Nothing
Dim SystemTimeIndex As String

For i As Integer = 0 To 4
    If Date.Compare(SystemTime, TList(i).TimeIn) >= 0
    And Date.Compare(SystemTime, TList(i).TimeOut) <= 0
    Then
        SystemTimeIndex = "1"
        SystemSked = SystemSked + SystemTimeIndex
    Else
        SystemTimeIndex = "0"
        SystemSked = SystemSked + SystemTimeIndex
    End If
Next

*****
*****
*****

*****
*****
*****

*****
*****
*****

'Check Existing Schedule

*****
*****
*****

If IdentifiedRoom <> Nothing Then
    Dim ds As New DataSet
    Dim da As New SqlDataAdapter("Select * from
tblSchedule where IDNumber = " & identifiedid & " AND
Room = " & IdentifiedRoom & " AND DayOfWeek = " &
day & "'", conn)
    conn.Open()
    da.Fill(ds)
    Dim Elist As SchedList()
    Dim list As DataRowCollection = ds.Tables(0).Rows
    ReDim Elist(list.Count)
    If list.Count = 0 Then
        conn.Close()
        Exit Sub
    End If
    For i As Integer = 1 To list.Count
        Elist(i).TimeIn = list.Item(i - 1).Item("TimeIn")
        Elist(i).TimeOut = list.Item(i - 1).Item("TimeOut")
        Elist(i).BinSked = list.Item(i - 1).Item("BinSked")
        Elist(i).TimeLength = list.Item(i -
1).Item("TimeLength")
    Next
    conn.Close()

    For i As Integer = 1 To Elist.Length
        For j As Integer = 0 To 4
            If Elist(i).BinSked(j) = "1" And SystemSked(j) =
"1" Then
                tbTimeIn = Elist(i).TimeIn
                tbTimeOut = Elist(i).TimeOut
                tbTimelength = Elist(i).TimeLength
                GoTo here
            End If
            Next
        Next
        Exit Sub
    End If

    here: Dim onethird As Integer = tbTimelength * 0.3
    Dim period As String = Nothing
    Dim str As String
    Dim str2 As String
    Dim hour As Integer
    Dim min As Integer
    Dim format As String

    format = tbTimeIn.Substring(5)
    str = tbTimeIn.Remove(1)
    hour = str
    str = tbTimeIn.Substring(2)
    str = str.Remove(2)
    min = str
    min = min + onethird
    If format = "PM" Then
        hour = hour + 12
    End If

    If min >= 60 Then
        hour = hour + 1
        min = min - 60
        str2 = min
        If hour > 12 Then
            hour = hour - 12
            str = hour
            period = str + ":" + str2 + " PM"
        Else
            str = hour
            period = str + ":" + str2 + " AM"
        End If
    ElseIf min < 60 Then
        str2 = min
        If min < 10 Then
            str2 = "0" + str2
        End If
        If hour > 12 Then
            hour = hour - 12
            str = hour
            period = str + ":" + str2 + " PM"
        Else
            str = hour
            period = str + ":" + str2 + " AM"
        End If
    End If

    If RoomStatus(IdentifiedRoom) = False Then
        If Date.Compare(SystemTime, tbTimeIn) >= 0
        And Date.Compare(SystemTime, period) <= 0 Then
            cmd = New SqlCommand("Update tblRooms Set
LockStatus = 1 where Room = " & IdentifiedRoom & "'",
conn)
            conn.Open()
            cmd.ExecuteNonQuery()
            conn.Close()

            display()
        End If
    End If

```

```

End If
Next
Next
Exit Sub

*****
*****
*****

*****
*****
*****

here: Dim onethird As Integer = tbTimelength * 0.3
Dim period As String = Nothing
Dim str As String
Dim str2 As String
Dim hour As Integer
Dim min As Integer
Dim format As String

format = tbTimeIn.Substring(5)
str = tbTimeIn.Remove(1)
hour = str
str = tbTimeIn.Substring(2)
str = str.Remove(2)
min = str
min = min + onethird
If format = "PM" Then
    hour = hour + 12
End If

If min >= 60 Then
    hour = hour + 1
    min = min - 60
    str2 = min
    If hour > 12 Then
        hour = hour - 12
        str = hour
        period = str + ":" + str2 + " PM"
    Else
        str = hour
        period = str + ":" + str2 + " AM"
    End If
ElseIf min < 60 Then
    str2 = min
    If min < 10 Then
        str2 = "0" + str2
    End If
    If hour > 12 Then
        hour = hour - 12
        str = hour
        period = str + ":" + str2 + " PM"
    Else
        str = hour
        period = str + ":" + str2 + " AM"
    End If
End If

If RoomStatus(IdentifiedRoom) = False Then
    If Date.Compare(SystemTime, tbTimeIn) >= 0
    And Date.Compare(SystemTime, period) <= 0 Then
        cmd = New SqlCommand("Update tblRooms Set
LockStatus = 1 where Room = " & IdentifiedRoom & "'",
conn)
        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()

        display()
    End If
End If

```

```

        frmExecute.txtTimeIn.Text =
Date.Now.ToShortTimeString

        ' SEND A SIGNAL PULSE TO UNLOCK THE
DOOR

        frmMain.MSComm.Open()
        frmMain.MSComm.Write(127)
        frmMain.MSComm.Close()

        dt = New DataTable
        da = New SqlDataAdapter("Select * from
tblDoorLock", conn)
        conn.Open()
        da.Fill(dt)
        cmb = New SqlCommandBuilder(da)
        newRow = dt.NewRow
        newRow("Room") = IdentifiedRoom
        newRow("TimeOut") = tbTimeOut
        newRow("LockAddress") = LockAddress
        dt.Rows.Add(newRow)
        da.InsertCommand = cmb.GetInsertCommand
        da.Update(dt)
        conn.Close()
    Else
        Exit Sub
    End If
    ElseIf RoomStatus(IdentifiedRoom) = True Then
        If Date.Compare(SystemTime, tbTimeIn) >= 0
And Date.Compare(SystemTime, tbTimeOut) <= 0 Then
            cmd = New SqlCommand("Update tblRooms Set
Lockstatus = 0 where Room = " & IdentifiedRoom & "'",
conn)
            conn.Open()
            cmd.ExecuteNonQuery()
            conn.Close()
            frmExecute.txtTimeOut.Text =
Date.Now.ToShortTimeString

            display()

            'SEND SIGNAL PULSE TO CLOSE THE
DOOR

            frmMain.MSComm.Open()
            frmMain.MSComm.Write(127)
            frmMain.MSComm.Close()
            accessrecord()

            cmd = New SqlCommand("Delete from
tblDoorLock where Room = " & IdentifiedRoom & "'",
conn)
            conn.Open()
            cmd.ExecuteNonQuery()
            conn.Close()
        Else
            Exit Sub
        End If
    End If
End If
End Sub

Private Function RoomStatus(ByVal IdentifiedRoom As
String) As Boolean
    Dim test As New getconnstring
    conn = New SqlConnection(test.getconn)
    Dim LockStatus As String = Nothing

    cmd = New SqlCommand("Select * from tblRooms
where Room = " & IdentifiedRoom & "'", conn)

    conn.Open()
    dr = cmd.ExecuteReader
    If dr.Read Then
        LockStatus = dr("LockStatus")
        dr.Close()
        conn.Close()
    Else
        dr.Close()
        conn.Close()
    End If

    If LockStatus = "0" Then
        Return False
    ElseIf LockStatus = "1" Then
        Return True
    End If
End Function

Private Sub display()
    Dim PFirstName As String
    Dim PMiddleName As String
    Dim PSurName As String
    Dim conn As SqlConnection
    Dim getConn As New getconnstring
    conn = New SqlConnection(getConn.getconn)

    cmd = New SqlCommand("Select * from tblTeach
where IDNumber = " & inputid & "'", conn)
    conn.Open()
    dr = cmd.ExecuteReader
    If dr.Read Then
        PFirstName = dr("Firstname")
        PMiddleName = dr("MiddleName")
        PSurName = dr("Surname")
        frmExecute.txtProfName.Text = PSurName + ", " +
PFirstName + " " + PMiddleName
        frmExecute.txtDepartment.Text = dr("Department")
        frmExecute.txtIDNumber.Text = inputid
        frmExecute.picProfessor.ImageLocation =
dr("ProfImage")
        dr.Close()
        conn.Close()
    Else
        dr.Close()
        conn.Close()
    End If

    cmd = New SqlCommand("Select * from tblSchedule
where IDNumber = " & inputid & " AND TimeIn = " &
tbtimein & " AND DayOfWeek = " &
Date.Today.DayOfWeek.ToString & "'", conn)
    conn.Open()
    dr = cmd.ExecuteReader
    If dr.Read Then
        frmExecute.txtSubject.Text = dr("SubjectName")
        frmExecute.txtRoom.Text = dr("Room")
        dr.Close()
        conn.Close()
    Else
        dr.Close()
        conn.Close()
    End If

    Dim ds As New DataSet

    da = New SqlDataAdapter("Select SubjectCode, Room,
DayOfWeek, TimeIn, TimeOut from tblSchedule", conn)
    conn.Open()
    da.Fill(ds, "Schedule")

```

```

        frmExecute.grdSchedule.DataSource =
ds.Tables("Schedule")
        conn.Close()

End Sub

Public Sub accessrecord()
    Dim conn As SqlConnection
    Dim cmb As SqlCommandBuilder
    Dim getConn As New getconnstring
    Dim dt As New DataTable
    Dim da As SqlDataAdapter

    conn = New SqlConnection(getConn.getconn)

    da = New SqlDataAdapter("Select * from
tblRoomAccess", conn)
    conn.Open()
    da.Fill(dt)
    cmb = New SqlCommandBuilder(da)

    Dim tmprow As DataRow
    tmprow = dt.NewRow
    tmprow("Num") = DBNull.Value
    tmprow("Name") = frmExecute.txtProfName.Text.Trim
    tmprow("Department") =
frmExecute.txtDepartment.Text.Trim
    tmprow("Subject") = frmExecute.txtSubject.Text.Trim
    tmprow("Recdate") = frmExecute.dtDate.Text.Trim
    tmprow("Timein") = frmExecute.txtTimeIn.Text.Trim
    tmprow("Timeout") =
frmExecute.txtTimeOut.Text.Trim
    tmprow("Room") = frmExecute.txtRoom.Text.Trim
    tmprow("IDNum") =
frmExecute.txtIDNumber.Text.Trim
    dt.Rows.Add(tmprow)

    da.InsertCommand = cmb.GetInsertCommand
    da.Update(dt)
    conn.Close()
    clearcontrols()
End Sub

Private Sub clearcontrols()
    frmExecute.picProfessor.ImageLocation = Nothing
    frmExecute.txtDepartment.Text = Nothing
    frmExecute.txtIDNumber.Text = Nothing
    frmExecute.txtProfName.Text = Nothing
    frmExecute.txtRoom.Text = Nothing
    frmExecute.txtSubject.Text = Nothing
    frmExecute.txtTimeIn.Text = Nothing
    frmExecute.txtTimeOut.Text = Nothing
    frmExecute.grdSchedule.DataSource = Nothing
End Sub
End Class

Util.vb

Imports GrFingerXLib
Imports Microsoft.VisualBasic

' Raw image data type.
Public Structure RawImage
    ' Image data.
    Public img As Object
    ' Image width.
    Public width As Long
    ' Image height.
    Public height As Long
    ' Image resolution.

```

```

    Public res As Long
End Structure

Public Class Util
    ' Some constants to make our code cleaner
    Public Const ERR_CANT_OPEN_BD As Integer = -999
    Public Const ERR_INVALID_ID As Integer = -998
    Public Const ERR_INVALID_TEMPLATE As Integer = -
997

    ' Importing necessary HDC functions
    Private Declare Function GetDC Lib "user32" (ByVal
hwnd As IntPtr) As IntPtr
    Private Declare Function ReleaseDC Lib "user32" (ByVal
hwnd As IntPtr, ByVal hdc As IntPtr) As IntPtr

    ' The last acquired image.
    Public raw As RawImage
    ' The template extracted from last acquired image.
    Public template As New TTemplate
    ' Database class.
    Public DB As DBClass
    ' Reference to main form log.
    Private _lbLog As ListBox
    ' Reference to main form Image.
    Private _pbPic As PictureBox
    ' GrFingerX component
    Private _GrFingerX As
AxGrFingerXLib.AxGrFingerXCtrl

    ' -----
    ' Support functions
    ' -----

    ' This class creates an Util class with some functions
    ' to help us to develop our GrFinger Application
    Public Sub New(ByRef lbLog As ListBox, ByRef pbPic
As PictureBox, ByRef GrFingerX As
AxGrFingerXLib.AxGrFingerXCtrl)
        _lbLog = lbLog
        _pbPic = pbPic
        _GrFingerX = GrFingerX
    End Sub

    ' Write a message in box.
    Public Sub WriteLog(ByVal message As String)
        _lbLog.Items.Add(message)
        _lbLog.SelectedIndex = _lbLog.Items.Count - 1
        _lbLog.ClearSelected()
    End Sub

    ' Write and describe an error.
    Public Sub WriteError(ByVal errorCode As Integer)
        Select Case errorCode
            Case GRConstants.GR_ERROR_INITIALIZE_FAIL
                WriteLog("Fail to Initialize GrFingerX. (Error:" &
errorCode & ")")
            Case
GRConstants.GR_ERROR_NOT_INITIALIZED
                WriteLog("The GrFingerX Library is not
initialized. (Error:" & errorCode & ")")
            Case
GRConstants.GR_ERROR_FAIL_LICENSE_READ
                WriteLog("License not found. See manual for
troubleshooting. (Error:" & errorCode & ")")
                MessageBox.Show("License not found. See
manual for troubleshooting.")

```

```

Case
GRConstants.GR_ERROR_NO_VALID_LICENSE
    WriteLog("The license is not valid. See manual for
troubleshooting. (Error:" & errorCode & ")")
    MessageBox.Show("The license is not valid. See
manual for troubleshooting.")
Case
GRConstants.GR_ERROR_NULL_ARGUMENT
    WriteLog("The parameter have a null value.
(Error:" & errorCode & ")")
Case GRConstants.GR_ERROR_FAIL
    WriteLog("Fail to create a GDI object. (Error:" &
errorCode & ")")
Case GRConstants.GR_ERROR_ALLOC
    WriteLog("Fail to create a context. Cannot allocate
memory. (Error:" & errorCode & ")")
Case GRConstants.GR_ERROR_PARAMETERS
    WriteLog("One or more parameters are out of
bound. (Error:" & errorCode & ")")
Case GRConstants.GR_ERROR_WRONG_USE
    WriteLog("This function cannot be called at this
time. (Error:" & errorCode & ")")
Case GRConstants.GR_ERROR_EXTRACT
    WriteLog("Template Extraction failed. (Error:" &
errorCode & ")")
Case
GRConstants.GR_ERROR_SIZE_OFF_RANGE
    WriteLog("Image is too larger or too short.
(Error:" & errorCode & ")")
Case GRConstants.GR_ERROR_RES_OFF_RANGE
    WriteLog("Image have too low or too high
resolution. (Error:" & errorCode & ")")
Case
GRConstants.GR_ERROR_CONTEXT_NOT_CREATED
    WriteLog("The Context could not be created.
(Error:" & errorCode & ")")
Case
GRConstants.GR_ERROR_INVALID_CONTEXT
    WriteLog("The Context does not exist. (Error:" &
errorCode & ")")

' Capture error codes

Case
GRConstants.GR_ERROR_CONNECT_SENSOR
    WriteLog("Error while connection to sensor.
(Error:" & errorCode & ")")
Case GRConstants.GR_ERROR_CAPTURING
    WriteLog("Error while capturing from sensor.
(Error:" & errorCode & ")")
Case
GRConstants.GR_ERROR_CANCEL_CAPTURING
    WriteLog("Error while stop capturing from sensor.
(Error:" & errorCode & ")")
Case
GRConstants.GR_ERROR_INVALID_ID_SENSOR
    WriteLog("The idSensor is invalid. (Error:" &
errorCode & ")")
Case
GRConstants.GR_ERROR_SENSOR_NOT_CAPTURING
    WriteLog("The sensor is not capturing. (Error:" &
errorCode & ")")
Case GRConstants.GR_ERROR_INVALID_EXT
    WriteLog("The File have a unknown extension.
(Error:" & errorCode & ")")
Case
GRConstants.GR_ERROR_INVALID_FILENAME
    WriteLog("The filename is invalid. (Error:" &
errorCode & ")")

Case
GRConstants.GR_ERROR_INVALID_FILETYPE
    WriteLog("The file type is invalid. (Error:" &
errorCode & ")")
Case GRConstants.GR_ERROR_SENSOR
    WriteLog("The sensor raise an error. (Error:" &
errorCode & ")")

' Our error codes

Case ERR_INVALID_TEMPLATE
    WriteLog("Invalid Template. (Error:" & errorCode
& ")")
Case ERR_INVALID_ID
    WriteLog("Invalid ID. (Error:" & errorCode & ")")
Case ERR_CANT_OPEN_BD
    WriteLog("Unable to connect to DataBase. (Error:"
& errorCode & ")")
Case Else
    WriteLog("Error:" & errorCode)
End Select
End Sub

' Check if we have a valid template
Private Function TemplateIsValid() As Boolean
' Check template size
Return template.Size > 0
End Function

' -----
' Main functions for fingerprint recognition management
' -----

' Initializes GrFinger ActiveX and all necessary utilities.
Public Function InitializeGrFinger() As Integer
Dim err As Integer

DB = New DBClass
' Open DataBase
If DB.OpenDB() = False Then Return
ERR_CANT_OPEN_BD
' Create a new Template
template.Size = 0
' Create a new raw image

DB.closeDB()

raw.img = Nothing
raw.width = 0
raw.height = 0
' Initializing library
err = _GrFingerX.Initialize()
If err < 0 Then Return err
Return _GrFingerX.CapInitialize()
End Function

' Finalizes and close the DB.
Public Sub FinalizeGrFinger()
' finalize library
_GrFingerX.Finalize()
_GrFingerX.CapFinalize()

' close DB
DB.closeDB()
DB = Nothing
End Sub

' Display fingerprint image on screen

```

```

Public Sub PrintBiometricDisplay(ByVal
biometricDisplay As Boolean, ByVal context As Integer)
    ' handle to finger image
    Dim handle As System.Drawing.Image = Nothing

    ' screen HDC
    Dim hdc As Integer = GetDC(0)

    If biometricDisplay Then
        ' get image with biometric info
        _GrFingerX.BiometricDisplay(template.tpt, raw.img,
raw.width, raw.height, raw.res, hdc, handle, context)
    Else
        ' get raw image
        _GrFingerX.CapRawImageToHandle(raw.img,
raw.width, raw.height, hdc, handle)
    End If

    ' draw image on picture box
    If Not (handle Is Nothing) Then
        _pbPic.Image = handle
        _pbPic.Update()
    End If
    ' release screen HDC
    ReleaseDC(0, hdc)
End Sub

' Add a fingerprint template to database
Public Function Enroll() As Integer
    ' Checking if template is valid.
    If TemplateIsValid() Then
        ' Adds template to database and gets ID.
        Return DB.AddTemplate(template)
    Else
        Return -1
    End If
End Function

' Extract a fingerprint template from current image
Function ExtractTemplate() As Integer
    Dim ret As Integer
    ' set current buffer size for extract template
    template.Size = template.tpt.Length

    ret = _GrFingerX.Extract(raw.img, raw.width,
raw.height, raw.res, template.tpt, template.Size,
GRConstants.GR_DEFAULT_CONTEXT)
    ' if error, set template size to 0
    ' Result < 0 => extraction problem
    If ret < 0 Then template.Size = 0
    Return ret
End Function

' Identify current fingerprint on our database
Public Function Identify(ByRef score As Integer) As
Integer
    Dim ret As Integer
    Dim i As Integer

    ' Checking if template is valid.
    If Not TemplateIsValid() Then Return
ERR_INVALID_TEMPLATE

    ' Starting identification process and supplying query
template.

    Dim tmpTpt As Array =
Array.CreateInstance(GetType(Byte), template.Size)
    Array.Copy(template.tpt, tmpTpt, template.Size)

```

```

    ret = _GrFingerX.IdentifyPrepare(tmpTpt,
GRConstants.GR_DEFAULT_CONTEXT)
    ' error?
    If ret < 0 Then Return ret
    ' Getting enrolled templates from database.
    Dim templates As TTemplates() = DB.getTemplates()
    ' Iterate over all templates in database
    For i = 1 To templates.Length
        ' Comparing the current template.
        If Not (templates(i - 1).template Is Nothing) Then
            Dim tempTpt As Array =
Array.CreateInstance(GetType(Byte), templates(i -
1).template.Size)
            Array.Copy(templates(i - 1).template.tpt, tempTpt,
templates(i - 1).template.Size)
            ret = _GrFingerX.Identify(tempTpt, score,
GRConstants.GR_DEFAULT_CONTEXT)
        End If
        ' Checking if query template and reference template
match.
        If ret = GRConstants.GR_MATCH Then
            Return templates(i - 1).ID
        End If
        If ret < 0 Then Return ret
    Next
    ' end of database, return "no match" code
    Return GRConstants.GR_NOT_MATCH
End Function
End Class

```

Variables.vb

```

Imports System.Data.SqlClient
Module variables

    Public dt As DataTable
    Public da As SqlDataAdapter
    Public dr As SqlDataReader
    Public cmd As SqlCommand
    Public cmb As SqlCommandBuilder
    Public newRow As DataRow

    Public adminuser As String
    Public profid As String
    Public sensorid As String
    Public inputid As String
    Public profile_timein As String

End Module

```

Source Doe for the Microcontroller

```

status                .equ 05h

com_flag              .equ 20h
reference             .equ 22h
data_byte            .equ 23h
ctr                  .equ 24h
bitrate              .equ 25h
shift_bit            .equ 26h
digit1               .equ 30h
digit2               .equ 31h
digit3               .equ 32h

.org 00h
.word 0ffffh;        ;p32
.word 0ffffh        ;p33
.word 0ffffh        ;p31
.word 0ffffh
.word baudrate
.word 0ffffh
.org 0ch

di                    ;disable interrupt
ld spl,#80h
ld p01m,#04h          ;port p0 as
input
ld p2m,#11111111b
                    ;port p2m as output
ld p3m,#01h
                    ;port p3m as input
srp #10h
                    ;register pointer
ld imr,#10h
clr irq
clr ipr
ld pre0,#00100101b;104 microsec
ld t0,#15;16 if using 11.150 mhz crystal, 15 if
using 10.240 mhz crysta
clr p0
clr p2
clr p3
clr 05h
call erase_ram
ei

main:
call erase_ram
ld r1,#27h
call rx_data
call hex_deci
ld r1,#27h

ld r15,@r1
cp digit1,r15
jr ne,invalid
inc r1
ld r15,@r1
cp digit2,r15
jr ne,invalid
inc r1
ld r15,@r1
cp digit3,r15
jr ne,invalid
call execute
jr main

invalid:
jr main

execute:
cp status,#0
jr eq,open
jp close

open:
or p0,#05h          ;solenoid
call delay2
and p0,#11111011b
or p0,#02h          ;open
loop_open:
tm p3,#04h
jr nz,loop_open
and p0,#11111100b
call delay
ld status,#0ffh
ret

close:
or p0,#01h
call delay2
or p0,#04h

loop_close:
tm p3,#02h
jr nz,loop_close
and p0,#11111010b
call delay
clr status
ret

rx_data:

```

```
tm p3,#08h
jr nz,rx_data
or com_flag,#01;
```

note:

D0 is used to differentiate rx and tx 1 indicate rx while 0 indicate tx;ld rx_flag,#0ffh

```
rx_data_loop
```

```
;or p0,#02h
```

```
tm p3,#08h
```

```
jr nz,rx_data_loop
```

```
ld tmr,#03h
```

```
clr data_byte
```

```
clr bitrate
```

```
and com_flag,#11111101b;D1 is used to set
communication flag ;clr com_flag
```

```
loop_incoming:
```

```
cp bitrate,#0
```

```
jr eq,loop_incoming
```

```
or com_flag,#02h;ld com_flag,#0ffh
```

```
check_byte:
```

```
cp ctr,#7
```

```
jr ne,check_byte
```

```
store_byte:
```

```
cp r1,#050h
```

```
jp eq,terminate
```

```
ld @r1,data_byte
```

```
inc r1
```

```
loop_stopbit:
```

```
cp ctr,#8 ;stop bit
```

```
jr ne,loop_stopbit
```

```
and tmr,#0fch
```

```
and com_flag,#11111101b;clr com_flag
```

```
clr ctr
```

```
ld r15,#0ffh
```

```
wait_fornext:
```

```
ld r14,#0ffh
```

```
loop_fornext:
```

```
tm p3,#08h
```

```
jp z,rx_data_loop
```

```
djnz r14,loop_fornext
```

```
djnz r15,wait_fornext
```

```
jr terminate
```

```
terminate:
```

```
and tmr,#0fch
```

```
and com_flag,#11111110b;clr rx_flag
```

```
ld @r1,data_byte
ret
```

```
tx_data:
```

```
rcf
```

```
or p0,#01h
```

```
clr ctr
```

```
clr bitrate
```

```
and com_flag,#11111110b;clr rx_flag
```

```
or tmr,#03h
```

```
idle: or p0,#01h ;idle
```

```
cp bitrate,#1
```

```
jr ne,idle
```

```
rl r0
```

```
rl r0
```

```
start_bit:
```

```
and p0,#0feh ;start bit
```

```
cp bitrate,#2
```

```
jr ne,start_bit
```

```
or p0,shift_bit;8 bit data shift
```

```
clr ctr
```

```
rcf
```

```
ld shift_bit,r0
```

```
and shift_bit,#01h
```

```
loop_upto8:
```

```
or p0,shift_bit ; data shift
```

```
cp ctr,#8
```

```
jr ult,loop_upto8
```

```
and tmr,#0fch
```

```
or p0,#01h
```

```
clr ctr
```

```
ret
```

```
nop
```

```
nop
```

```
nop
```

```
baudrate:
```

```
tm com_flag,#01h;cp rx_flag,#0
```

```
jr nz,recieve
```

```
transmit:
```

```
rr r0
```

```
ld shift_bit,r0
```

```
and shift_bit,#01h
```

```
and p0,#0feh
```

```
inc ctr
```

```
inc bitrate
```

```
iret
```

```

recieve:
tm com_flag,#02h;cp com_flag,#0
jr z,junk_startbit
swap shift_bit
and shift_bit,#80h
or data_byte,shift_bit
rr data_byte
inc ctr
ld shift_bit,p3
iret
junk_startbit:
ld shift_bit,p3
inc bitrate
iret

time_expired:
or 03h,#08h
iret

hex_deci:
ld r15,p2

hundred:
cp r15,#100
jr ult,tens
sub r15,#100
inc digit1
jr hundred

tens:
cp r15,#10
jr ult,ones
sub r15,#10
inc digit2
jr tens

ones:
ld digit3,r15
or digit1,#30h
or digit2,#30h
or digit3,#30h
ret

;xxxxxxdelay routine xxxxxxxxxxxx

delay:ld r3,#01fh

loop1:
ld r2,#0ffh

loop2:
djnz r2,loop2
djnz r3,loop1

ret

delay2:
ld r0,#50

del:
call delay
dec r0
cp r0,#0
jr ne,del
ret

delayk:
ld r6,p0
and r6,#07h
cp r6,#07h
jr ne,delayk
ret

erase_ram:
ld r0,#00
ld r1,#20h

clean2:
ld @r1,r0
inc r1
cp r1,#060h
jr ne,clean2
ld r1,#31h
ret

.end

```

APPENDIX D
MAX 232 DATASHEET

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where $\pm 12V$ is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5 μ W. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers
Low-Power Modems
Interface Translation
Battery-Powered RS-232 Systems
Multidrop RS-232 Networks

Features

Superior to Bipolar

- ◆ Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)
- ◆ Low-Power Receive Mode in Shutdown (MAX223/MAX242)
- ◆ Meet All EIA/TIA-232E and V.28 Specifications
- ◆ Multiple Drivers and Receivers
- ◆ 3-State Driver and Receiver Outputs
- ◆ Open-Line Detection (MAX243)

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (μ F)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✓	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✓	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✓	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✓	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✓	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✓	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✓	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✓	120	Available in quad flatpack package

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC})	-0.3V to +6V	20-Pin Plastic DIP (derate 8.00mW/°C above +70°C)	..440mW
Input Voltages		16-Pin Narrow SO (derate 8.70mW/°C above +70°C)	...696mW
T_{IN}	-0.3V to ($V_{CC} - 0.3V$)	16-Pin Wide SO (derate 9.52mW/°C above +70°C)762mW
R_{IN} (Except MAX220) $\pm 30V$	18-Pin Wide SO (derate 9.52mW/°C above +70°C)762mW
R_{IN} (MAX220) $\pm 25V$	20-Pin Wide SO (derate 10.00mW/°C above +70°C)800mW
T_{OUT} (Except MAX220) (Note 1) $\pm 15V$	20-Pin SSOP (derate 8.00mW/°C above +70°C)640mW
T_{OUT} (MAX220) $\pm 13.2V$	16-Pin CERDIP (derate 10.00mW/°C above +70°C)800mW
Output Voltages		18-Pin CERDIP (derate 10.53mW/°C above +70°C)842mW
T_{OUT} $\pm 15V$	Operating Temperature Ranges	
R_{OUT}-0.3V to ($V_{CC} + 0.3V$)	MAX2_ _AC_ _ , MAX2_ _C_ _0°C to +70°C
Driver/Receiver Output Short Circuited to GNDContinuous	MAX2_ _AE_ _ , MAX2_ _E_ _-40°C to +85°C
Continuous Power Dissipation ($T_A = +70^\circ C$)		MAX2_ _AM_ _ , MAX2_ _M_ _-55°C to +125°C
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C)842mW	Storage Temperature Range-65°C to +160°C
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C)889mW	Lead Temperature (soldering, 10s)+300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, \overline{SHDN} or $V_{CC} = 0V$.

Note 2: For the MAX220, V_+ and V_- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

($V_{CC} = +5V \pm 10\%$, C_1 – $C_4 = 0.1\mu F$, MAX220, $C_1 = 0.047\mu F$, C_2 – $C_4 = 0.33\mu F$, $T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V _{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	μA
	SHDN = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V _{CC} = 5.5V, SHDN = 0V, V _{OUT} = ±15V, MAX222/242			±0.01	±10	μA
	V _{CC} = SHDN = 0V, V _{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kbps
Transmitter Output Resistance	V _{CC} = V ₊ = V ₋ = 0V, V _{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V _{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V _{CC} = 5V	All except MAX243 R _{2IN}	0.8	1.3		V
		MAX243 R _{2IN} (Note 2)	-3			
RS-232 Input Threshold High	V _{CC} = 5V	All except MAX243 R _{2IN}		1.8	2.4	V
		MAX243 R _{2IN} (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V _{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I _{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I _{OUT} = -1.0mA		3.5	V _{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V _{OUT} = GND		-2	-10		mA
	Sinking V _{OUT} = V _{CC}		10	30		

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1μF, MAX220, C1 = 0.047μF, C2–C4 = 0.33μF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

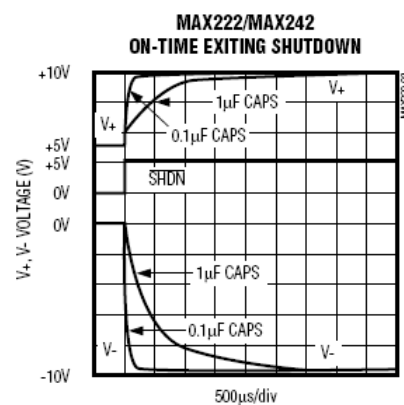
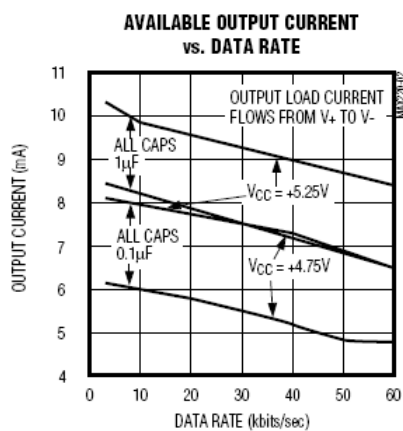
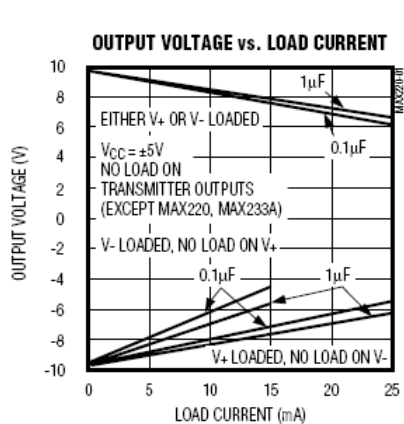
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{\text{SHDN}} = V_{CC}$ or $\overline{\text{EN}} = V_{CC}$ ($\overline{\text{SHDN}} = 0V$ for MAX222), $0V \leq V_{OUT} \leq V_{CC}$			±0.05	±10	μA
$\overline{\text{EN}}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{\text{EN}}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V _{CC} Supply Current ($\overline{\text{SHDN}} = V_{CC}$), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T _A = +25°C		0.1	10	μA
		T _A = 0°C to +70°C		2	50	
		T _A = -40°C to +85°C		2	50	
		T _A = -55°C to +125°C		35	100	
$\overline{\text{SHDN}}$ Input Leakage Current	MAX222/242				±1	μA
$\overline{\text{SHDN}}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{\text{SHDN}}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C _L = 50pF to 2500pF, R _L = 3kΩ to 7kΩ, V _{CC} = 5V, T _A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/μs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (Normal Operation), Figure 1	t _{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	μs
		MAX220		4	10	
	t _{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (Normal Operation), Figure 2	t _{PHLR}	MAX222/232A/233A/242/243		0.5	1	μs
		MAX220		0.6	3	
	t _{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (Shutdown), Figure 2	t _{PHLS}	MAX242		0.5	10	μs
	t _{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t _{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t _{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{\text{SHDN}}$ Goes High), Figure 4	t _{ET}	MAX222/242, 0.1μF caps (includes charge-pump start-up)		250		μs
Transmitter-Output Disable Time ($\overline{\text{SHDN}}$ Goes Low), Figure 4	t _{DT}	MAX222/242, 0.1μF caps		600		ns
Transmitter + to - Propagation Delay Difference (Normal Operation)	t _{PHLT} - t _{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (Normal Operation)	t _{PHLR} - t _{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 3: MAX243 R2_{OUT} is guaranteed to be low when R2_{IN} is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243



+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX223/MAX230–MAX241

V _{CC}	-0.3V to +6V	20-Pin Wide SO (derate 10.00mW/°C above +70°C).....	800mW
V ₊	(V _{CC} - 0.3V) to +14V	24-Pin Wide SO (derate 11.76mW/°C above +70°C).....	941mW
V ₋	+0.3V to -14V	28-Pin Wide SO (derate 12.50mW/°C above +70°C).....	1W
Input Voltages			
T _{IN}	-0.3V to (V _{CC} + 0.3V)	44-Pin Plastic FP (derate 11.11mW/°C above +70°C).....	889mW
R _{IN}	±30V	14-Pin Cerdip (derate 9.09mW/°C above +70°C).....	727mW
Output Voltages			
T _{OUT}	(V ₊ + 0.3V) to (V ₋ - 0.3V)	16-Pin Cerdip (derate 10.00mW/°C above +70°C).....	800mW
R _{OUT}	-0.3V to (V _{CC} + 0.3V)	20-Pin Cerdip (derate 11.11mW/°C above +70°C).....	889mW
Short-Circuit Duration, T _{OUT}			
Continuous Power Dissipation (T _A = +70°C)			
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C).....	800mW	24-Pin Narrow Cerdip (derate 12.50mW/°C above +70°C).....	1W
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C).....	842mW	24-Pin Sidebrazed (derate 20.0mW/°C above +70°C).....	1.6W
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C).....	889mW	28-Pin SSOP (derate 9.52mW/°C above +70°C).....	762mW
24-Pin Narrow Plastic DIP (derate 13.33mW/°C above +70°C).....	1.07W	Operating Temperature Ranges	
24-Pin Plastic DIP (derate 9.09mW/°C above +70°C).....	500mW	MAX2 __ C	0°C to +70°C
16-Pin Wide SO (derate 9.52mW/°C above +70°C).....	762mW	MAX2 __ E	-40°C to +85°C
		MAX2 __ M	-55°C to +125°C
		Storage Temperature Range	
		Lead Temperature (soldering, 10s)	

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241

(MAX223/230/232/234/236/237/238/240/241, V_{CC} = +5V ±10%; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0μF; MAX231/MAX239, V_{CC} = 5V ±10%; V₊ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to ground		±5.0	±7.3		V
V _{CC} Power-Supply Current	No load, T _A = +25°C	MAX232/233		5	10	mA
		MAX223/230/234–238/240/241		7	15	
		MAX231/239		0.4	1	
V ₊ Power-Supply Current		MAX231		1.8	5	mA
		MAX239		5	15	
Shutdown Supply Current	T _A = +25°C	MAX223		15	50	μA
		MAX230/235/236/240/241		1	10	
Input Logic Threshold Low	T _{IN} ; EN, $\overline{\text{SHDN}}$ (MAX233); $\overline{\text{EN}}$, SHDN (MAX230/235–241)				0.8	V
Input Logic Threshold High	T _{IN}		2.0			V
	EN, $\overline{\text{SHDN}}$ (MAX223); $\overline{\text{EN}}$, SHDN (MAX230/235/236/240/241)		2.4			
Logic Pull-Up Current	T _{IN} = 0V			1.5	200	μA
Receiver Input Voltage Operating Range			-30		30	V

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241 (continued)

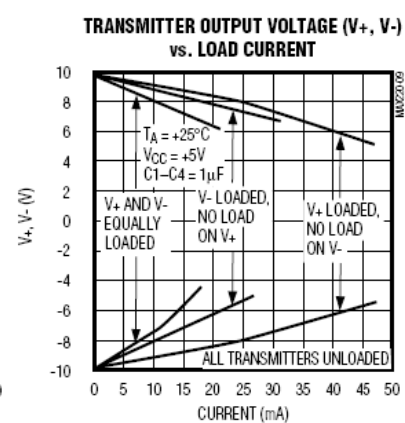
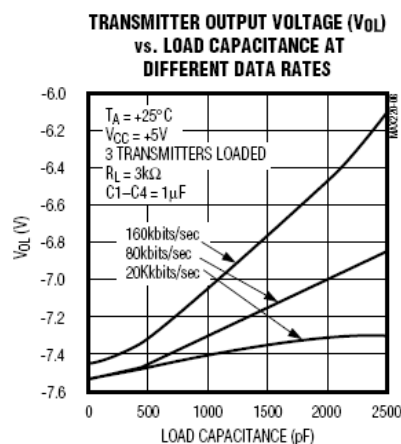
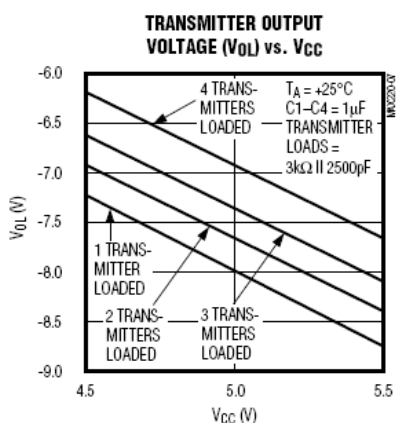
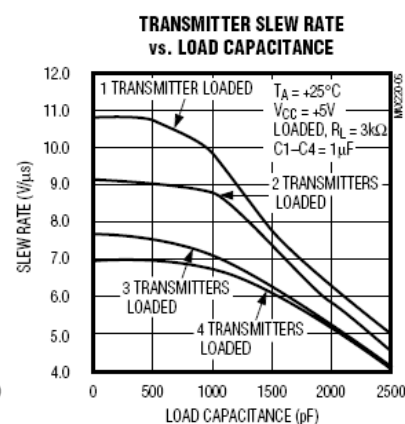
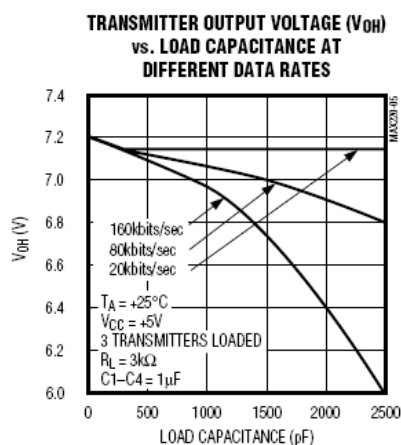
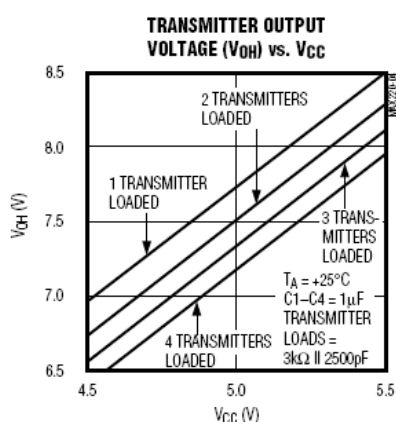
(MAX223/230/232/234/236/237/238/240/241, $V_{CC} = +5V \pm 10\%$; MAX233/MAX235, $V_{CC} = 5V \pm 5\%$, C_1 – $C_4 = 1.0\mu F$; MAX231/MAX239, $V_{CC} = 5V \pm 10\%$; $V_+ = 7.5V$ to $13.2V$; $T_A = T_{MIN}$ to T_{MAX} ; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 Input Threshold Low	$T_A = +25^\circ C$, $V_{CC} = 5V$	Normal operation SHDN = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)	0.8	1.2		V
		Shutdown (MAX223) SHDN = 0V, EN = 5V (R_{4IN} , R_{5IN})	0.6	1.5		
RS-232 Input Threshold High	$T_A = +25^\circ C$, $V_{CC} = 5V$	Normal operation SHDN = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)		1.7	2.4	V
		Shutdown (MAX223) SHDN = 0V, EN = 5V (R_{4IN} , R_{5IN})		1.5	2.4	
RS-232 Input Hysteresis	$V_{CC} = 5V$, no hysteresis in shutdown		0.2	0.5	1.0	V
RS-232 Input Resistance	$T_A = +25^\circ C$, $V_{CC} = 5V$		3	5	7	$k\Omega$
TTL/CMOS Output Voltage Low	$I_{OUT} = 1.6mA$ (MAX231/232/233, $I_{OUT} = 3.2mA$)				0.4	V
TTL/CMOS Output Voltage High	$I_{OUT} = -1mA$		3.5	$V_{CC} - 0.4$		V
TTL/CMOS Output Leakage Current	$0V \leq R_{OUT} \leq V_{CC}$; EN = 0V (MAX223); EN = V_{CC} (MAX235–241)			0.05	± 10	μA
Receiver Output Enable Time	Normal operation	MAX223		600		ns
		MAX235/236/239/240/241		400		
Receiver Output Disable Time	Normal operation	MAX223		900		ns
		MAX235/236/239/240/241		250		
Propagation Delay	RS-232 IN to TTL/CMOS OUT, $C_L = 150pF$	Normal operation		0.5	10	μs
		SHDN = 0V (MAX223)	t_{PHLS}	4	40	
			t_{PLHS}	6	40	
Transition Region Slew Rate	MAX223/MAX230/MAX234–241, $T_A = +25^\circ C$, $V_{CC} = 5V$, $R_L = 3k\Omega$ to $7k\Omega$, $C_L = 50pF$ to $2500pF$, measured from +3V to -3V or -3V to +3V		3	5.1	30	V/ μs
	MAX231/MAX232/MAX233, $T_A = +25^\circ C$, $V_{CC} = 5V$, $R_L = 3k\Omega$ to $7k\Omega$, $C_L = 50pF$ to $2500pF$, measured from +3V to -3V or -3V to +3V			4	30	
Transmitter Output Resistance	$V_{CC} = V_+ = V_- = 0V$, $V_{OUT} = \pm 2V$		300			Ω
Transmitter Output Short-Circuit Current				± 10		mA

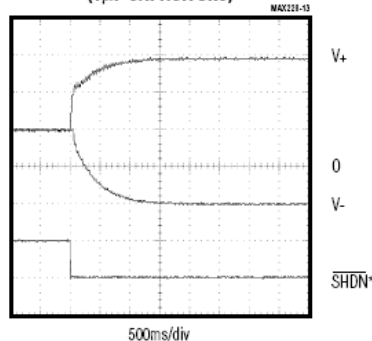
+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX223/MAX230-MAX241



V_+ , V_- WHEN EXITING SHUTDOWN (1µF CAPACITORS)



*SHUTDOWN POLARITY IS REVERSED FOR NON MAX241 PARTS

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX225/MAX244—MAX249

Supply Voltage (V_{CC})	-0.3V to +6V	Continuous Power Dissipation ($T_A = +70^\circ\text{C}$)	
Input Voltages		28-Pin Wide SO (derate 12.50mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$)	1W
T_{IN} , \overline{ENA} , \overline{ENB} , \overline{ENR} , \overline{ENT} , \overline{ENRA} , \overline{ENRB} , \overline{ENTA} , \overline{ENTB}	-0.3V to ($V_{CC} + 0.3\text{V}$)	40-Pin Plastic DIP (derate 11.11mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$)	611mW
R_{IN}	$\pm 25\text{V}$	44-Pin PLCC (derate 13.33mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$)	1.07W
T_{OUT} (Note 3)	$\pm 15\text{V}$	Operating Temperature Ranges	
R_{OUT}	-0.3V to ($V_{CC} + 0.3\text{V}$)	MAX225C_-, MAX24_C_-	0°C to $+70^\circ\text{C}$
Short Circuit (one output at a time)		MAX225E_-, MAX24_E_-	-40°C to $+85^\circ\text{C}$
T_{OUT} to GND	Continuous	Storage Temperature Range	-65°C to $+160^\circ\text{C}$
R_{OUT} to GND	Continuous	Lead Temperature (soldering, 10s)	$+300^\circ\text{C}$

Note 4: Input voltage measured with transmitter output in a high-impedance state, shutdown, or $V_{CC} = 0\text{V}$.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX225/MAX244—MAX249

(MAX225, $V_{CC} = 5.0\text{V} \pm 5\%$; MAX244—MAX249, $V_{CC} = +5.0\text{V} \pm 10\%$, external capacitors C1–C4 = $1\mu\text{F}$; $T_A = T_{MIN}$ to T_{MAX} ; unless otherwise noted.)

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS					
Input Logic Threshold Low			1.4	0.8	V
Input Logic Threshold High		2	1.4		V
Logic Pull-Up/Input Current	Tables 1a–1d	Normal operation		10	μA
		Shutdown		± 0.01	
Data Rate	Tables 1a–1d, normal operation		120	64	kbps
Output Voltage Swing	All transmitter outputs loaded with $3\text{k}\Omega$ to GND	± 5	± 7.5		V
Output Leakage Current (Shutdown)	Tables 1a–1d	\overline{ENA} , \overline{ENB} , \overline{ENT} , \overline{ENTA} , $\overline{ENTB} = V_{CC}$, $V_{OUT} = \pm 15\text{V}$		± 0.01	μA
		$V_{CC} = 0\text{V}$, $V_{OUT} = \pm 15\text{V}$		± 0.01	
Transmitter Output Resistance	$V_{CC} = V_+ = V_- = 0\text{V}$, $V_{OUT} = \pm 2\text{V}$ (Note 4)	300	10M		Ω
Output Short-Circuit Current	$V_{OUT} = 0\text{V}$	± 7	± 30		mA
RS-232 RECEIVERS					
RS-232 Input Voltage Operating Range				± 25	V
RS-232 Input Threshold Low	$V_{CC} = 5\text{V}$	0.8	1.3		V
RS-232 Input Threshold High	$V_{CC} = 5\text{V}$		1.8	2.4	V
RS-232 Input Hysteresis	$V_{CC} = 5\text{V}$	0.2	0.5	1.0	V
RS-232 Input Resistance		3	5	7	$\text{k}\Omega$
TTL/CMOS Output Voltage Low	$I_{OUT} = 3.2\text{mA}$		0.2	0.4	V
TTL/CMOS Output Voltage High	$I_{OUT} = -1.0\text{mA}$	3.5	$V_{CC} - 0.2$		V
TTL/CMOS Output Short-Circuit Current	Sourcing $V_{OUT} = \text{GND}$	-2	-10		mA
	Shrinking $V_{OUT} = V_{CC}$	10	30		
TTL/CMOS Output Leakage Current	Normal operation, outputs disabled, Tables 1a–1d, $0\text{V} \leq V_{OUT} \leq V_{CC}$, $\overline{ENR}_- = V_{CC}$		± 0.05	± 0.10	μA

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX225/MAX244–MAX249 (continued)

(MAX225, $V_{CC} = 5.0V \pm 5\%$; MAX244–MAX249, $V_{CC} = +5.0V \pm 10\%$, external capacitors C1–C4 = $1\mu F$; $T_A = T_{MIN}$ to T_{MAX} ; unless otherwise noted.)

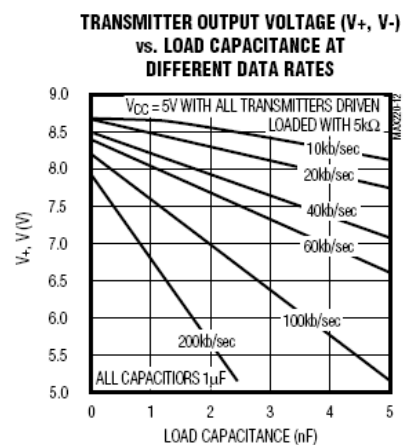
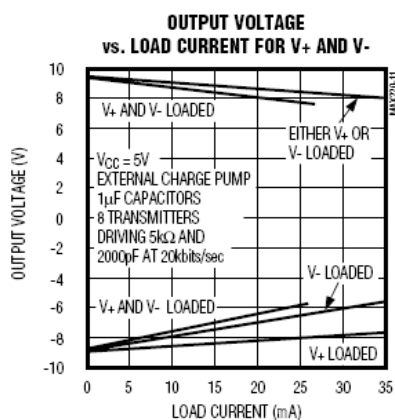
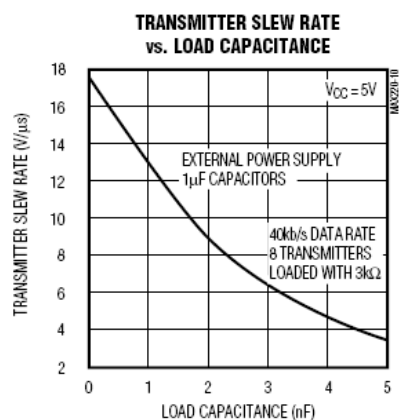
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY AND CONTROL LOGIC						
Operating Supply Voltage		MAX225	4.75	5.25	V	
		MAX244–MAX249	4.5	5.5		
V _{CC} Supply Current (Normal Operation)	No load	MAX225	10	20	mA	
		MAX244–MAX249	11	30		
	3kΩ loads on all outputs	MAX225	40			
		MAX244–MAX249	57			
Shutdown Supply Current	T _A = +25°C		8	25	μA	
	T _A = T _{MIN} to T _{MAX}			50		
Control Input	Leakage current			±1	μA	
	Threshold low		1.4	0.8	V	
	Threshold high		2.4	1.4		
AC CHARACTERISTICS						
Transition Slew Rate	C _L = 50pF to 2500pF, R _L = 3kΩ to 7kΩ, V _{CC} = 5V, T _A = +25°C, measured from +3V to -3V or -3V to +3V		5	10	30	V/μs
Transmitter Propagation Delay TLL to RS-232 (Normal Operation), Figure 1	t _{PHLT}		1.3	3.5	μs	
	t _{PLHT}		1.5	3.5		
Receiver Propagation Delay TLL to RS-232 (Normal Operation), Figure 2	t _{PHLR}		0.6	1.5	μs	
	t _{PLHR}		0.6	1.5		
Receiver Propagation Delay TLL to RS-232 (Low-Power Mode), Figure 2	t _{PHLS}		0.6	10	μs	
	t _{PLHS}		3.0	10		
Transmitter + to - Propagation Delay Difference (Normal Operation)	t _{PHLT} - t _{PLHT}		350		ns	
Receiver + to - Propagation Delay Difference (Normal Operation)	t _{PHLR} - t _{PLHR}		350		ns	
Receiver-Output Enable Time, Figure 3	t _{ER}		100	500	ns	
Receiver-Output Disable Time, Figure 3	t _{DR}		100	500	ns	
Transmitter Enable Time	t _{ET}	MAX246–MAX249 (excludes charge-pump startup)	5		μs	
		MAX225/MAX245–MAX249 (includes charge-pump startup)	10		ms	
Transmitter Disable Time, Figure 4	t _{DT}		100		ns	

Note 5: The 300 Ω minimum specification complies with EIA/TIA-232E, but the actual resistance when in shutdown mode or $V_{CC} = 0V$ is 10M Ω as is implied by the leakage specification.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX225/MAX244-MAX249



+5V-Powered, Multichannel RS-232 Drivers/Receivers

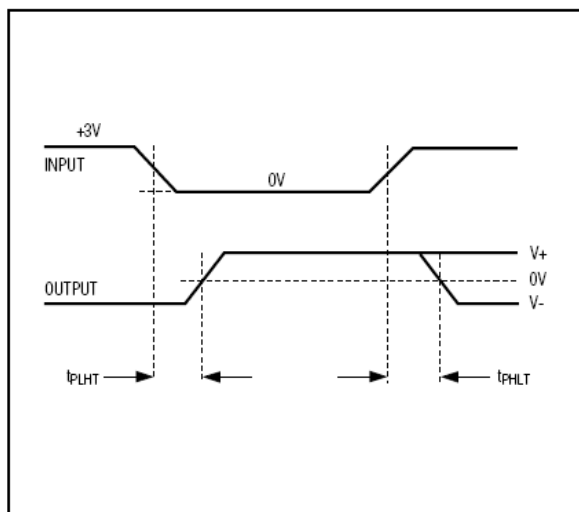


Figure 1. Transmitter Propagation-Delay Timing

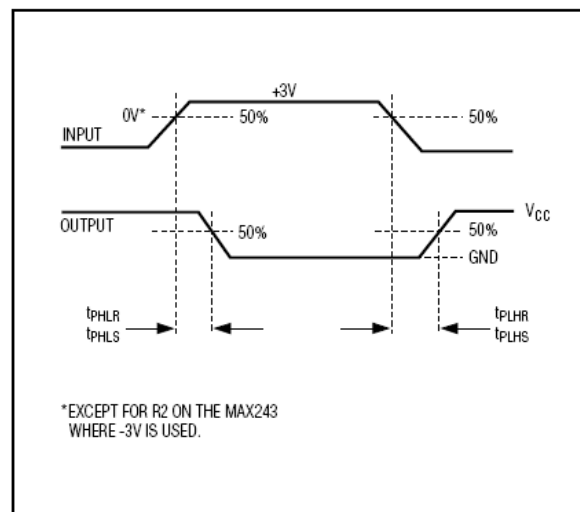


Figure 2. Receiver Propagation-Delay Timing

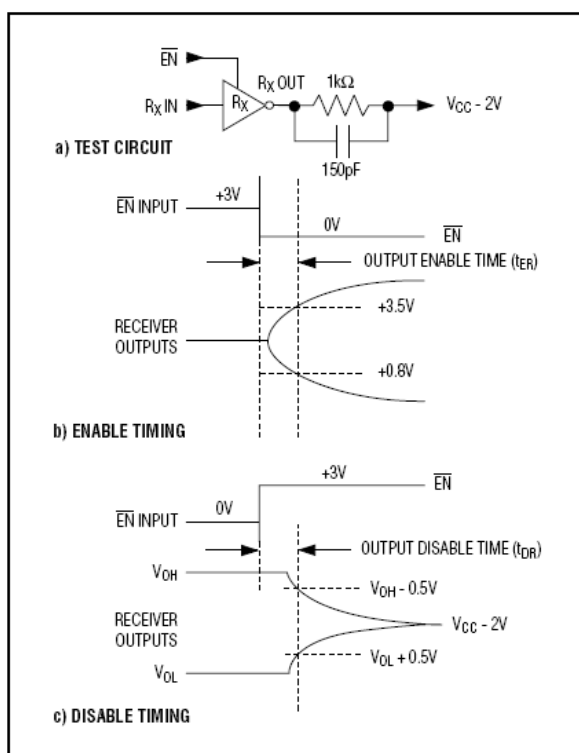


Figure 3. Receiver-Output Enable and Disable Timing

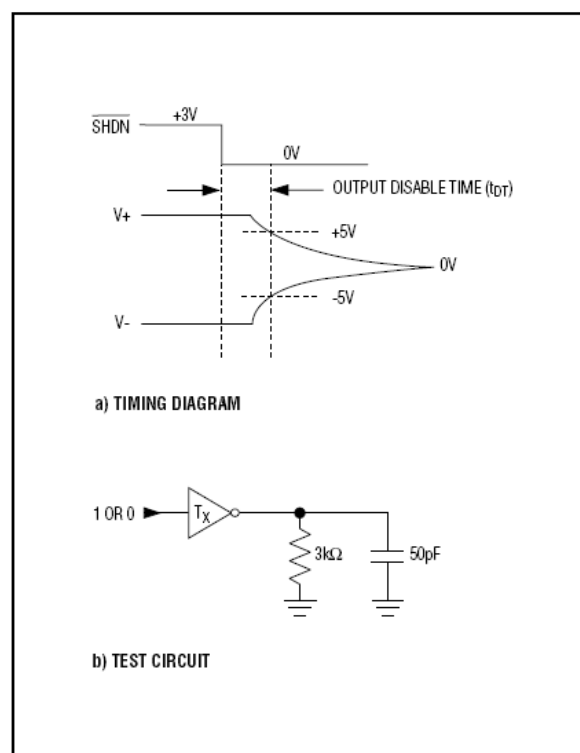


Figure 4. Transmitter-Output Disable Timing

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Table 1a. MAX245 Control Pin Configurations

$\overline{\text{ENT}}$	$\overline{\text{ENR}}$	OPERATION STATUS	TRANSMITTERS	RECEIVERS
0	0	Normal Operation	All Active	All Active
0	1	Normal Operation	All Active	All 3-State
1	0	Shutdown	All 3-State	All Low-Power Receive Mode
1	1	Shutdown	All 3-State	All 3-State

Table 1b. MAX245 Control Pin Configurations

$\overline{\text{ENT}}$	$\overline{\text{ENR}}$	OPERATION STATUS	TRANSMITTERS		RECEIVERS	
			TA1–TA4	TB1–TB4	RA1–RA5	RB1–RB5
0	0	Normal Operation	All Active	All Active	All Active	All Active
0	1	Normal Operation	All Active	All Active	RA1–RA4 3-State, RA5 Active	RB1–RB4 3-State, RB5 Active
1	0	Shutdown	All 3-State	All 3-State	All Low-Power Receive Mode	All Low-Power Receive Mode
1	1	Shutdown	All 3-State	All 3-State	RA1–RA4 3-State, RA5 Low-Power Receive Mode	RB1–RB4 3-State, RB5 Low-Power Receive Mode

Table 1c. MAX246 Control Pin Configurations

$\overline{\text{ENA}}$	$\overline{\text{ENB}}$	OPERATION STATUS	TRANSMITTERS		RECEIVERS	
			TA1–TA4	TB1–TB4	RA1–RA5	RB1–RB5
0	0	Normal Operation	All Active	All Active	All Active	All Active
0	1	Normal Operation	All Active	All 3-State	All Active	RB1–RB4 3-State, RB5 Active
1	0	Shutdown	All 3-State	All Active	RA1–RA4 3-State, RA5 Active	All Active
1	1	Shutdown	All 3-State	All 3-State	RA1–RA4 3-State, RA5 Low-Power Receive Mode	RB1–RB4 3-State, RA5 Low-Power Receive Mode

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Table 1d. MAX247/MAX248/MAX249 Control Pin Configurations

<u>ENT</u> A	<u>ENT</u> B	<u>EN</u> RA	<u>EN</u> RB	OPERATION STATUS	TRANSMITTERS			RECEIVERS	
					MAX247	TA1–TA4	TB1–TB4	RA1–RA4	RB1–RB5
					MAX248	TA1–TA4	TB1–TB4	RA1–RA4	RB1–RB4
					MAX249	TA1–TA3	TB1–TB3	RA1–RA5	RB1–RB5
0	0	0	0	Normal Operation		All Active	All Active	All Active	All Active
0	0	0	1	Normal Operation		All Active	All Active	All Active	All 3-State, except RB5 stays active on MAX247
0	0	1	0	Normal Operation		All Active	All Active	All 3-State	All Active
0	0	1	1	Normal Operation		All Active	All Active	All 3-State	All 3-State, except RB5 stays active on MAX247
0	1	0	0	Normal Operation		All Active	All 3-State	All Active	All Active
0	1	0	1	Normal Operation		All Active	All 3-State	All Active	All 3-State, except RB5 stays active on MAX247
0	1	1	0	Normal Operation		All Active	All 3-State	All 3-State	All Active
0	1	1	1	Normal Operation		All Active	All 3-State	All 3-State	All 3-State, except RB5 stays active on MAX247
1	0	0	0	Normal Operation		All 3-State	All Active	All Active	All Active
1	0	0	1	Normal Operation		All 3-State	All Active	All Active	All 3-State, except RB5 stays active on MAX247
1	0	1	0	Normal Operation		All 3-State	All Active	All 3-State	All Active
1	0	1	1	Normal Operation		All 3-State	All Active	All 3-State	All 3-State, except RB5 stays active on MAX247
1	1	0	0	Shutdown		All 3-State	All 3-State	Low-Power Receive Mode	Low-Power Receive Mode
1	1	0	1	Shutdown		All 3-State	All 3-State	Low-Power Receive Mode	All 3-State, except RB5 stays active on MAX247
1	1	1	0	Shutdown		All 3-State	All 3-State	All 3-State	Low-Power Receive Mode
1	1	1	1	Shutdown		All 3-State	All 3-State	All 3-State	All 3-State, except RB5 stays active on MAX247

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Detailed Description

The MAX220–MAX249 contain four sections: dual charge-pump DC-DC voltage converters, RS-232 drivers, RS-232 receivers, and receiver and transmitter enable control inputs.

Dual Charge-Pump Voltage Converter

The MAX220–MAX249 have two internal charge-pumps that convert +5V to $\pm 10V$ (unloaded) for RS-232 driver operation. The first converter uses capacitor C1 to double the +5V input to +10V on C3 at the V+ output. The second converter uses capacitor C2 to invert +10V to -10V on C4 at the V- output.

A small amount of power may be drawn from the +10V (V+) and -10V (V-) outputs to power external circuitry (see the *Typical Operating Characteristics* section), except on the MAX225 and MAX245–MAX247, where these pins are not available. V+ and V- are not regulated, so the output voltage drops with increasing load current. Do not load V+ and V- to a point that violates the minimum $\pm 5V$ EIA/TIA-232E driver output voltage when sourcing current from V+ and V- to external circuitry.

When using the shutdown feature in the MAX222, MAX225, MAX230, MAX235, MAX236, MAX240, MAX241, and MAX245–MAX249, avoid using V+ and V- to power external circuitry. When these parts are shut down, V- falls to 0V, and V+ falls to +5V. For applications where a +10V external supply is applied to the V+ pin (instead of using the internal charge pump to generate +10V), the C1 capacitor must not be installed and the $\overline{\text{SHDN}}$ pin must be tied to VCC. This is because V+ is internally connected to VCC in shutdown mode.

RS-232 Drivers

The typical driver output voltage swing is $\pm 8V$ when loaded with a nominal 5k Ω RS-232 receiver and VCC = +5V. Output swing is guaranteed to meet the EIA/TIA-232E and V.28 specification, which calls for $\pm 5V$ minimum driver output levels under worst-case conditions. These include a minimum 3k Ω load, VCC = +4.5V, and maximum operating temperature. Unloaded driver output voltage ranges from (V+ -1.3V) to (V- +0.5V).

Input thresholds are both TTL and CMOS compatible. The inputs of unused drivers can be left unconnected since 400k Ω input pull-up resistors to VCC are built in (except for the MAX220). The pull-up resistors force the outputs of unused drivers low because all drivers invert. The internal input pull-up resistors typically source 12 μA , except in shutdown mode where the pull-ups are disabled. Driver outputs turn off and enter a high-impedance state—where leakage current is typically microamperes (maximum 25 μA)—when in shutdown

mode, in three-state mode, or when device power is removed. Outputs can be driven to $\pm 15V$. The power-supply current typically drops to 8 μA in shutdown mode. The MAX220 does not have pull-up resistors to force the outputs of the unused drivers low. Connect unused inputs to GND or VCC.

The MAX239 has a receiver three-state control line, and the MAX223, MAX225, MAX235, MAX236, MAX240, and MAX241 have both a receiver three-state control line and a low-power shutdown control. Table 2 shows the effects of the shutdown control and receiver three-state control on the receiver outputs.

The receiver TTL/CMOS outputs are in a high-impedance, three-state mode whenever the three-state enable line is high (for the MAX225/MAX235/MAX236/MAX239–MAX241), and are also high-impedance whenever the shutdown control line is high.

When in low-power shutdown mode, the driver outputs are turned off and their leakage current is less than 1 μA with the driver output pulled to ground. The driver output leakage remains less than 1 μA , even if the transmitter output is backdriven between 0V and (VCC + 6V). Below -0.5V, the transmitter is diode clamped to ground with 1k Ω series impedance. The transmitter is also zener clamped to approximately VCC + 6V, with a series impedance of 1k Ω .

The driver output slew rate is limited to less than 30V/ μs as required by the EIA/TIA-232E and V.28 specifications. Typical slew rates are 24V/ μs unloaded and 10V/ μs loaded with 3 Ω and 2500pF.

RS-232 Receivers

EIA/TIA-232E and V.28 specifications define a voltage level greater than 3V as a logic 0, so all receivers invert. Input thresholds are set at 0.8V and 2.4V, so receivers respond to TTL level inputs as well as EIA/TIA-232E and V.28 levels.

The receiver inputs withstand an input overvoltage up to $\pm 25V$ and provide input terminating resistors with

Table 2. Three-State Control of Receivers

PART	SHDN	$\overline{\text{SHDN}}$	EN	$\overline{\text{EN}}(\text{R})$	RECEIVERS
MAX223	—	Low High High	X Low High	—	High Impedance Active High Impedance
MAX225	—	—	—	Low High	High Impedance Active
MAX235 MAX236 MAX240	Low Low High	—	—	Low High X	High Impedance Active High Impedance

+5V-Powered, Multichannel RS-232 Drivers/Receivers

nominal $5k\Omega$ values. The receivers implement Type 1 interpretation of the fault conditions of V.28 and EIA/TIA-232E.

The receiver input hysteresis is typically 0.5V with a guaranteed minimum of 0.2V. This produces clear output transitions with slow-moving input signals, even with moderate amounts of noise and ringing. The receiver propagation delay is typically 600ns and is independent of input swing direction.

Low-Power Receive Mode

The low-power receive-mode feature of the MAX223, MAX242, and MAX245–MAX249 puts the IC into shutdown mode but still allows it to receive information. This is important for applications where systems are periodically awakened to look for activity. Using low-power receive mode, the system can still receive a signal that will activate it on command and prepare it for communication at faster data rates. This operation conserves system power.

Negative Threshold—MAX243

The MAX243 is pin compatible with the MAX232A, differing only in that RS-232 cable fault protection is removed on one of the two receiver inputs. This means that control lines such as CTS and RTS can either be driven or left floating without interrupting communication. Different cables are not needed to interface with different pieces of equipment.

The input threshold of the receiver without cable fault protection is -0.8V rather than +1.4V. Its output goes positive only if the input is connected to a control line that is actively driven negative. If not driven, it defaults to the 0 or “OK to send” state. Normally, the MAX243’s other receiver (+1.4V threshold) is used for the data line (TD or RD), while the negative threshold receiver is connected to the control line (DTR, DTS, CTS, RTS, etc.).

Other members of the RS-232 family implement the optional cable fault protection as specified by EIA/TIA-232E specifications. This means a receiver output goes high whenever its input is driven negative, left floating, or shorted to ground. The high output tells the serial communications IC to stop sending data. To avoid this, the control lines must either be driven or connected with jumpers to an appropriate positive voltage level.

Shutdown—MAX222–MAX242

On the MAX222, MAX235, MAX236, MAX240, and MAX241, all receivers are disabled during shutdown. On the MAX223 and MAX242, two receivers continue to operate in a reduced power mode when the chip is in shutdown. Under these conditions, the propagation delay increases to about 2.5 μ s for a high-to-low input transition. When in shutdown, the receiver acts as a CMOS inverter with no hysteresis. The MAX223 and MAX242 also have a receiver output enable input ($\overline{\text{EN}}$ for the MAX242 and EN for the MAX223) that allows receiver output control independent of SHDN (SHDN for MAX241). With all other devices, $\overline{\text{SHDN}}$ (SHDN for MAX241) also disables the receiver outputs.

The MAX225 provides five transmitters and five receivers, while the MAX245 provides ten receivers and eight transmitters. Both devices have separate receiver and transmitter-enable controls. The charge pumps turn off and the devices shut down when a logic high is applied to the ENT input. In this state, the supply current drops to less than 25 μ A and the receivers continue to operate in a low-power receive mode. Driver outputs enter a high-impedance state (three-state mode). On the MAX225, all five receivers are controlled by the $\overline{\text{ENR}}$ input. On the MAX245, eight of the receiver outputs are controlled by the $\overline{\text{ENR}}$ input, while the remaining two receivers (RA5 and RB5) are always active. RA1–RA4 and RB1–RB4 are put in a three-state mode when $\overline{\text{ENR}}$ is a logic high.

Receiver and Transmitter Enable Control Inputs

The MAX225 and MAX245–MAX249 feature transmitter and receiver enable controls.

The receivers have three modes of operation: full-speed receive (normal active), three-state (disabled), and low-power receive (enabled receivers continue to function at lower data rates). The receiver enable inputs control the full-speed receive and three-state modes. The transmitters have two modes of operation: full-speed transmit (normal active) and three-state (disabled). The transmitter enable inputs also control the shutdown mode. The device enters shutdown mode when all transmitters are disabled. Enabled receivers function in the low-power receive mode when in shutdown.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Tables 1a–1d define the control states. The MAX244 has no control pins and is not included in these tables.

The MAX246 has ten receivers and eight drivers with two control pins, each controlling one side of the device. A logic high at the A-side control input ($\overline{\text{ENA}}$) causes the four A-side receivers and drivers to go into a three-state mode. Similarly, the B-side control input ($\overline{\text{ENB}}$) causes the four B-side drivers and receivers to go into a three-state mode. As in the MAX245, one A-side and one B-side receiver (RA5 and RB5) remain active at all times. The entire device is put into shutdown mode when both the A and B sides are disabled ($\overline{\text{ENA}} = \overline{\text{ENB}} = +5\text{V}$).

The MAX247 provides nine receivers and eight drivers with four control pins. The $\overline{\text{ENRA}}$ and $\overline{\text{ENRB}}$ receiver enable inputs each control four receiver outputs. The $\overline{\text{ENTA}}$ and $\overline{\text{ENTB}}$ transmitter enable inputs each control four drivers. The ninth receiver (RB5) is always active. The device enters shutdown mode with a logic high on both $\overline{\text{ENTA}}$ and $\overline{\text{ENTB}}$.

The MAX248 provides eight receivers and eight drivers with four control pins. The $\overline{\text{ENRA}}$ and $\overline{\text{ENRB}}$ receiver enable inputs each control four receiver outputs. The $\overline{\text{ENTA}}$ and $\overline{\text{ENTB}}$ transmitter enable inputs control four drivers each. This part does not have an always-active receiver. The device enters shutdown mode and transmitters go into a three-state mode with a logic high on both $\overline{\text{ENTA}}$ and $\overline{\text{ENTB}}$.

The MAX249 provides ten receivers and six drivers with four control pins. The $\overline{\text{ENRA}}$ and $\overline{\text{ENRB}}$ receiver enable inputs each control five receiver outputs. The $\overline{\text{ENTA}}$ and $\overline{\text{ENTB}}$ transmitter enable inputs control three drivers each. There is no always-active receiver. The device enters shutdown mode and transmitters go into a three-state mode with a logic high on both $\overline{\text{ENTA}}$ and $\overline{\text{ENTB}}$. In shutdown mode, active receivers operate in a low-power receive mode at data rates up to 20kbits/sec.

Applications Information

Figures 5 through 25 show pin configurations and typical operating circuits. In applications that are sensitive to power-supply noise, VCC should be decoupled to ground with a capacitor of the same value as C1 and C2 connected as close as possible to the device.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

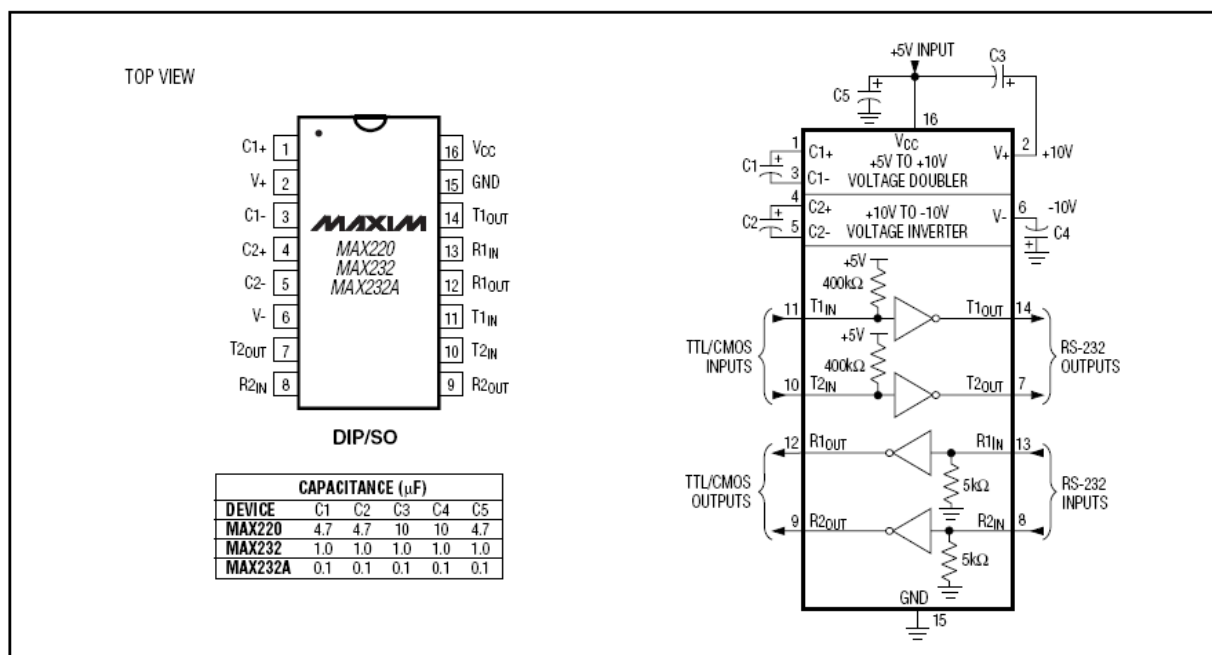


Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

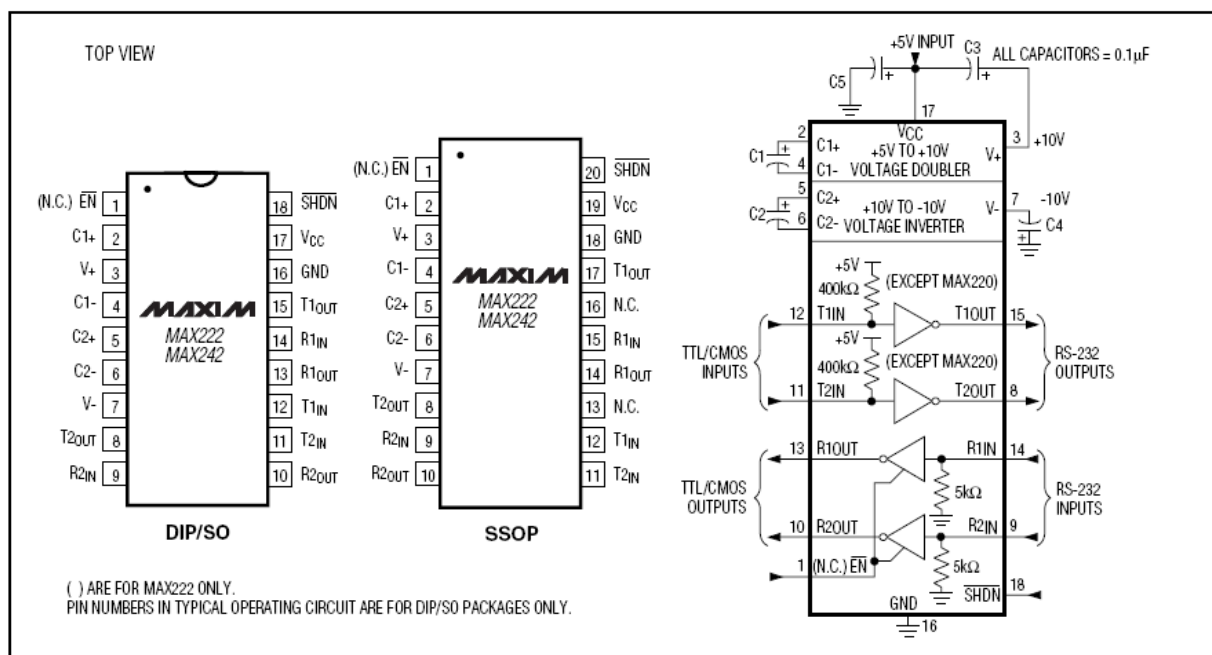


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

APPENDIX E
Z86E08 DATASHEET



Z86E04/E08

CMOS Z8 OTP MICROCONTROLLERS

PRODUCT DEVICES

Part Number	Oscillator Type	Operating V_{CC}	Operating Temperature	ROM (KB)	Package
Z86E0412PEC	Crystal	4.5V–5.5V	–40°C/105°C	1	18-Pin DIP
Z86E0412PSC1866	Crystal	4.5V–5.5V	0°C/70°C	1	18-Pin DIP
Z86E0412PSC1903	RC	4.5V–5.5V	0°C/70°C	1	18-Pin DIP
Z86E0412PEC1903	RC	4.5V–5.5V	–40°C/105°C	1	18-Pin DIP
Z86E0412SEC	Crystal	4.5V–5.5V	–40°C/105°C	1	18-Pin SOIC
Z86E0412SSC1866	Crystal	4.5V–5.5V	0°C/70°C	1	18-Pin SOIC
Z86E0412SSC1903	RC	4.5V–5.5V	0°C/70°C	1	18-Pin SOIC
Z86E0412SEC1903	RC	4.5V–5.5V	–40°C/105°C	1	18-Pin SOIC
Z86E0812PEC	Crystal	4.5V–5.5V	–40°C/105°C	2	18-Pin DIP
Z86E0812PSC1866	Crystal	4.5V–5.5V	0°C/70°C	2	18-Pin DIP
Z86E0812PSC1903	RC	4.5V–5.5V	0°C/70°C	2	18-Pin DIP
Z86E0812PEC1903	RC	4.5V–5.5V	–40°C/105°C	2	18-Pin DIP
Z86E0812SEC	Crystal	4.5V–5.5V	–40°C/105°C	2	18-Pin SOIC
Z86E0812SSC1866	Crystal	4.5V–5.5V	0°C/70°C	2	18-Pin SOIC
Z86E0812SSC1903	RC	4.5V–5.5V	0°C/70°C	2	18-Pin SOIC
Z86E0812SEC1903	RC	4.5V–5.5V	–40°C/105°C	2	18-Pin SOIC

Several key product features of the extensive family of Zilog Z86E04/E08 CMOS OTP microcontrollers are presented in the above table. This table enables the user to identify which of the E04/E08 product variants most closely match the user's application requirements.

FEATURES

- 14 Input/Output Lines
- Six Vectored, Prioritized Interrupts (3 falling edge, 1 rising edge, 2 timers)
- Two Analog Comparators
- Program Options:
 - Low Noise
 - ROM Protect
 - Auto Latch
 - Watch-Dog Timer (WDT)
 - EPROM/Test Mode Disable
- Two Programmable 8-Bit Counter/Timers, Each with 6-Bit Programmable Prescaler
- WDT/ Power-On Reset (POR)
- On-Chip Oscillator that Accepts XTAL, Ceramic Resonance, LC, RC, or External Clock
- Clock-Free WDT Reset
- Low-Power Consumption (50 mw typical)
- Fast Instruction Pointer (1μs @ 12 MHz)
- RAM Bytes (125)

GENERAL DESCRIPTION

Zilog's Z86E04/E08 Microcontrollers (MCU) are One-Time Programmable (OTP) members of Zilog's single-chip Z8® MCU family that allow easy software development, debug, prototyping, and small production runs not economically desirable with masked ROM versions.

For applications demanding powerful I/O capabilities, the Z86E04/E08's dedicated input and output lines are grouped into three ports, and are configurable under software control to provide timing, status signals, or parallel I/O.

Two on-chip counter/timers, with a large number of user selectable modes, offload the system of administering real-time tasks such as counting/timing and I/O data communications.

Note: All Signals with an overline, " $\overline{}$ ", are active Low, for example: $\overline{B/W}$ (WORD is active Low); \overline{B}/W (BYTE is active Low, only).

Power connections follow conventional descriptions below:

Connection	Circuit	Device
Power	V_{CC}	V_{DD}
Ground	GND	V_{SS}

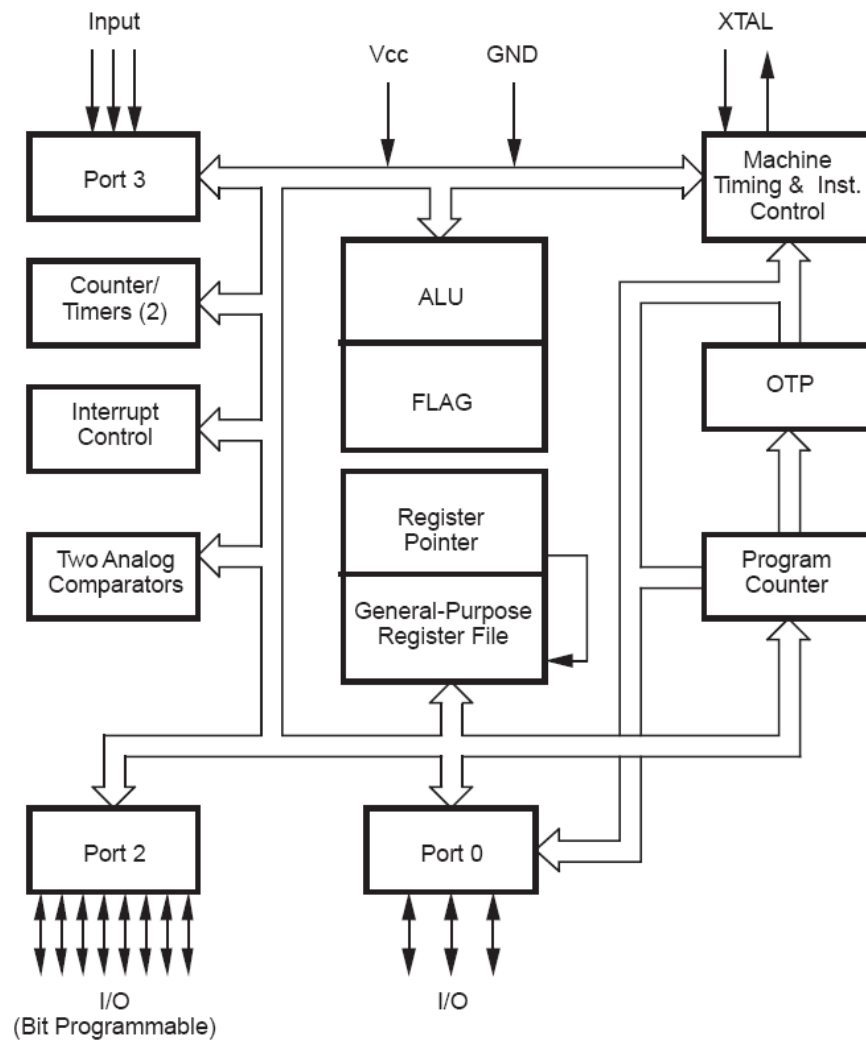
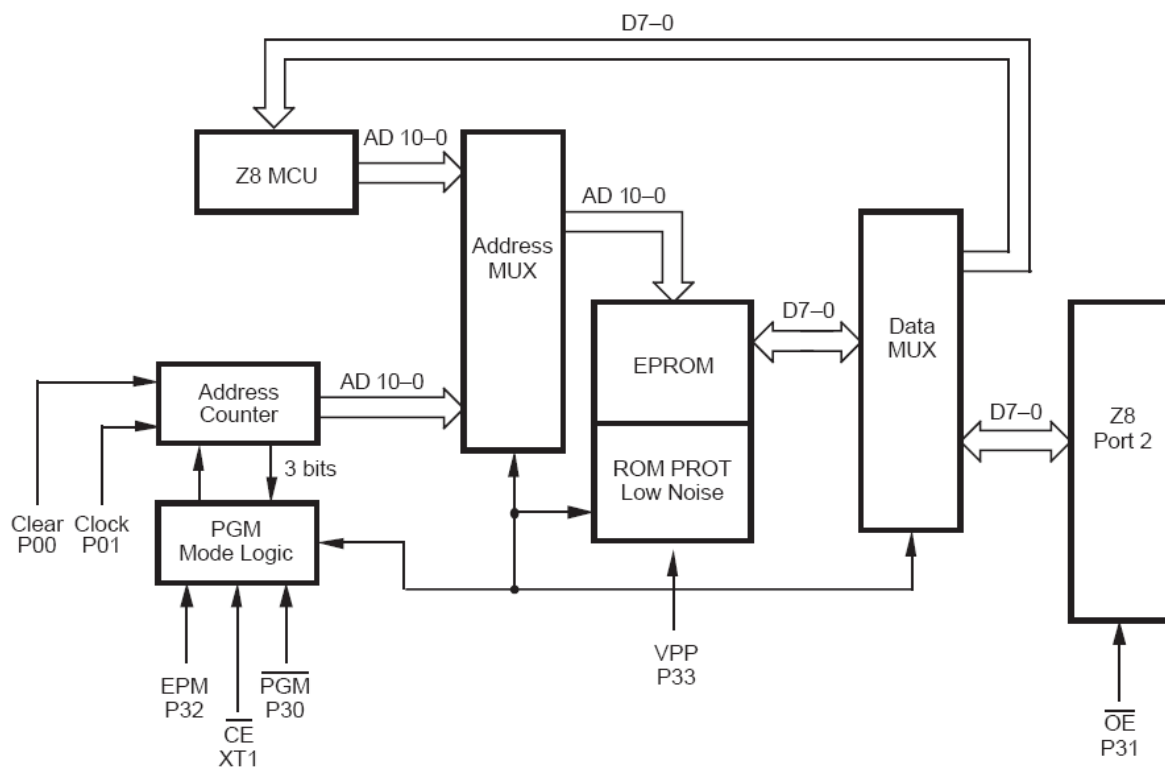


Figure 1. Functional Block Diagram

GENERAL DESCRIPTION (Continued)**Figure 2. EPROM Programming Mode Block Diagram**

PIN DESCRIPTION

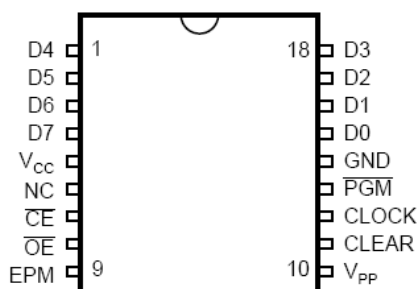


Figure 3. 18-Pin EPROM Mode Configuration

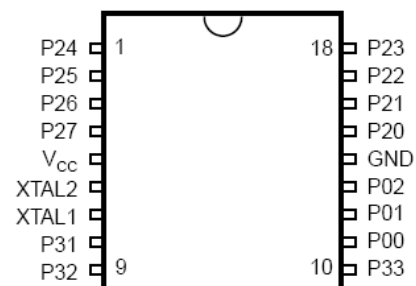


Figure 4. 18-Pin DIP/SOIC Mode Configuration

Table 1. 18-Pin DIP Pin Identification

EPROM Programming Mode			
Pin #	Symbol	Function	Direction
1–4	D4–D7	Data 4, 5, 6, 7	In/Output
5	V _{CC}	Power Supply	
6	NC	No Connection	
7	CE	Chip Enable	Input
8	OE	Output Enable	Input
9	EPM	EPROM Prog Mode	Input
10	V _{PP}	Prog Voltage	Input
11	Clear	Clear Clock	Input
12	Clock	Address	Input
13	PGM	Prog Mode	Input
14	GND	Ground	
15–18	D0–D3	Data 0,1, 2, 3	In/Output

Table 2. 18-Pin DIP/SOIC Pin Identification

Standard Mode			
Pin #	Symbol	Function	Direction
1–4	P24–P27	Port 2, Pins 4,5,6,7	In/Output
5	V _{CC}	Power Supply	
6	XTAL2	Crystal Osc. Clock	Output
7	XTAL1	Crystal Osc. Clock	Input
8	P31	Port 3, Pin 1, AN1	Input
9	P32	Port 3, Pin 2, AN2	Input
10	P33	Port 3, Pin 3, REF	Input
11–13	P00–P02	Port 0, Pins 0,1,2	In/Output
14	GND	Ground	
15–18	P20–P23	Port 2, Pins 0,1,2,3	In/Output

ABSOLUTE MAXIMUM RATINGS

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for an extended period may affect device reliability. Total power

dissipation should not exceed 462 mW for the package. Power dissipation is calculated as follows:

$$\begin{aligned} \text{Total Power Dissipation} = & V_{DD} \times [I_{DD} - (\text{sum of } I_{OH})] \\ & + \text{sum of } [(V_{DD} - V_{OH}) \times I_{OH}] \\ & + \text{sum of } (V_{OL} \times I_{OL}) \end{aligned}$$

Parameter	Min	Max	Units	Note
Ambient Temperature under Bias	-40	+105	C	
Storage Temperature	-65	+150	C	
Voltage on any Pin with Respect to V_{SS}	-0.7	+12	V	1
Voltage on V_{DD} Pin with Respect to V_{SS}	-0.3	+7	V	
Voltage on Pins 7, 8, 9, 10 with Respect to V_{SS}	-0.6	$V_{DD}+1$	V	2
Total Power Dissipation		1.65	W	
Maximum Allowable Current out of V_{SS}		300	mA	
Maximum Allowable Current into V_{DD}		220	mA	
Maximum Allowable Current into an Input Pin	-600	+600	μ A	3
Maximum Allowable Current into an Open-Drain Pin	-600	+600	μ A	4
Maximum Allowable Output Current Sunked by Any I/O Pin		25	mA	
Maximum Allowable Output Current Sourced by Any I/O Pin		25	mA	
Total Maximum Output Current Sunked by a Port		60	mA	
Total Maximum Output Current Sourced by a Port		45	mA	

Notes:

1. This applies to all pins except where otherwise noted. Maximum current into pin must be $\pm 600 \mu$ A.
2. There is no input protection diode from pin to V_{DD} (not applicable to EPROM Mode).
3. This excludes Pin 6 and Pin 7.
4. Device pin is not at an output Low state.

STANDARD TEST CONDITIONS

The characteristics listed below apply for standard test conditions as noted. All voltages are referenced to Ground. Positive current flows into the referenced pin (Figure 5).

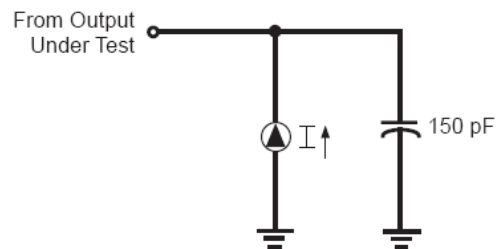


Figure 5. Test Load Diagram

CAPACITANCE

$T_A = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0\text{V}$, $f = 1.0\text{ MHz}$, unmeasured pins returned to GND.

Parameter	Min	Max
Input capacitance	0	10 pF
Output capacitance	0	20 pF
I/O capacitance	0	25 pF

DC ELECTRICAL CHARACTERISTICS

Standard Temperature

Sym	Parameter	V _{CC} [4]	T _A = 0°C to +70°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max				
V _{INMAX}	Max Input Voltage	4.5V		12		V	I _{IN} < 250 μA	1
		5.5V		12		V	I _{IN} < 250 μA	1
V _{CH}	Clock Input High Voltage	4.5V	0.8 V _{CC}	V _{CC} +0.3	2.8	V	Driven by External Clock Generator	
		5.5V	0.8 V _{CC}	V _{CC} +0.3	2.8	V	Driven by External Clock Generator	
V _{CL}	Clock Input Low Voltage	4.5V	V _{SS} -0.3	0.2 V _{CC}	1.7	V	Driven by External Clock Generator	
		5.5V	V _{SS} -0.3	0.2 V _{CC}	1.7	V	Driven by External Clock Generator	
V _{IH}	Input High Voltage	4.5V	0.7 V _{CC}	V _{CC} +0.3	2.8	V		
		5.5V	0.7 V _{CC}	V _{CC} +0.3	2.8	V		
V _{IL}	Input Low Voltage	4.5V	V _{SS} -0.3	0.2 V _{CC}	1.5	V		
		5.5V	V _{SS} -0.3	0.2 V _{CC}	1.5	V		
V _{OH}	Output High Voltage	4.5V	V _{CC} -0.4		4.8	V	I _{OH} = -2.0 mA	5
		5.5V	V _{CC} -0.4		4.8	V	I _{OH} = -2.0 mA	5
		4.5V	V _{CC} -0.4		4.8	V	Low Noise @ I _{OH} = -0.5 mA	
		5.5V	V _{CC} -0.4		4.8	V	Low Noise @ I _{OH} = -0.5 mA	
V _{OL1}	Output Low Voltage	4.5V		0.8	0.1	V	I _{OL} = +4.0 mA	5
		5.5V		0.4	0.1	V	I _{OL} = +4.0 mA	5
		4.5V		0.4	0.1	V	Low Noise @ I _{OL} = 1.0 mA	
		5.5V		0.4	0.1	V	Low Noise @ I _{OL} = 1.0 mA	
V _{OL2}	Output Low Voltage	4.5V		0.8	0.8	V	I _{OL} = +12 mA,	5
		5.5V		0.8	0.8	V	I _{OL} = +12 mA,	5
V _{OFFSET}	Comparator Input Offset Voltage	4.5V		25.0	10.0	mV		
		5.5V		25.0	10.0	mV		
V _{LV}	V _{CC} Low Voltage Auto Reset		2.2	3.0	2.8	V	@ 6 MHz Max. Int. CLK Freq.	
I _{IL}	Input Leakage (Input Bias Current of Comparator)	4.5V	-1.0	1.0		μA	V _{IN} = 0V, V _{CC}	
		5.5V	-1.0	1.0		μA	V _{IN} = 0V, V _{CC}	
I _{OL}	Output Leakage	4.5V	-1.0	1.0		μA	V _{IN} = 0V, V _{CC}	
		5.5V	-1.0	1.0		μA	V _{IN} = 0V, V _{CC}	
V _{ICR}	Comparator Input Common Mode Voltage Range		0	V _{CC} -1.0		V		

Sym	Parameter	V _{CC} [4]	T _A = 0°C to +70°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max				
I _{CC}	Supply Current	4.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 2 MHz	5,7
		5.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 2 MHz	5,7
		4.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 8 MHz	5,7
		5.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 8 MHz	5,7
		4.5V		20.0	12.0	mA	All Output and I/O Pins Floating @ 12 MHz	5,7
		5.5V		20.0	12.0	mA	All Output and I/O Pins Floating @ 12 MHz	5,7
I _{CC1}	Standby Current	4.5V		4.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	5,7
		5.5V		4.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	5,7
		4.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 8 MHz	5,7
		5.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 8 MHz	5,7
		4.5V		7.0	4.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 12 MHz	5,7
		5.5V		7.0	4.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 12 MHz	5,7
I _{CC}	Supply Current (Low Noise Mode)	4.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 1 MHz	7
		5.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 1 MHz	7
		4.5V		13.0	7.5	mA	All Output and I/O Pins Floating @ 2 MHz	7
		5.5V		13.0	7.5	mA	All Output and I/O Pins Floating @ 2 MHz	7
		4.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 4 MHz	7
		5.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 4 MHz	7

DC ELECTRICAL CHARACTERISTICS (Continued)

Sym	Parameter	V _{CC} [4]	T _A = 0°C to +70°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max				
I _{CC1}	Standby Current (Low Noise Mode)	4.5V		4.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 1 MHz	7
		5.5V		4.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 1 MHz	7
		4.5V		4.5	2.8	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	7
		5.5V		4.5	2.8	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	7
		4.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 4 MHz	7
		5.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 4 MHz	7
I _{CC2}	Standby Current	4.5V		10.0	1.0	μA	STOP Mode V _{IN} = 0V, V _{CC} WDT is not Running	7,8
		5.5V		10.0	1.0	μA	STOP Mode V _{IN} = 0V, V _{CC} WDT is not Running	7,8
I _{ALL}	Auto Latch Low Current	4.5V		32.0	16	μA	0V < V _{IN} < V _{CC}	
		5.5V		32.0	16	μA	0V < V _{IN} < V _{CC}	
I _{ALH}	Auto Latch High Current	4.5V		-16.0	-8.0	μA	0V < V _{IN} < V _{CC}	
		5.5V		-16.0	-8.0	μA	0V < V _{IN} < V _{CC}	

Notes:

1. Port 2 and Port 0 only
2. V_{SS} = 0V = GND
3. The device operates down to V_{LV} of the specified frequency for V_{LV}. The minimum operational V_{CC} is determined on the value of the voltage V_{LV} at the ambient temperature. The V_{LV} increases as the temperature decreases.
4. V_{CC} = 4.5 to 5.5V, typical values measured at V_{CC} = 5.0V.
The V_{CC} voltage specification of 5.5 V guarantees 5.0 V ± 0.5V with typical values measured at V_{CC} = 5.0V.
5. Standard Mode (not Low EMI Mode)
6. Z86E08 only
7. All outputs unloaded and all inputs are at V_{CC} or V_{SS} level.
8. If analog comparator is selected, then the comparator inputs must be at V_{CC} level.

DC ELECTRICAL CHARACTERISTICS

Extended Temperature

Sym	Parameter	V _{CC} [4]	T _A = -40°C to +105°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max				
V _{INMAX}	Max Input Voltage	4.5V		12.0		V	I _{IN} < 250 µA	1
		5.5V		12.0		V	I _{IN} < 250 µA	1
V _{CH}	Clock Input High Voltage	4.5V	0.8 V _{CC}	V _{CC} +0.3	2.8	V	Driven by External Clock Generator	
		5.5V	0.8 V _{CC}	V _{CC} +0.3	2.8	V	Driven by External Clock Generator	
V _{CL}	Clock Input Low Voltage	4.5V	V _{SS} -0.3	0.2 V _{CC}	1.7	V	Driven by External Clock Generator	
		5.5V	V _{SS} -0.3	0.2 V _{CC}	1.7	V	Driven by External Clock Generator	
V _{IH}	Input High Voltage	4.5V	0.7 V _{CC}	V _{CC} +0.3	2.8	V		
		5.5V	0.7 V _{CC}	V _{CC} +0.3	2.8	V		
V _{IL}	Input Low Voltage	4.5V	V _{SS} -0.3	0.2 V _{CC}	1.5	V		
		5.5V	V _{SS} -0.3	0.2 V _{CC}	1.5	V		
V _{OH}	Output High Voltage	4.5V	V _{CC} -0.4		4.8	V	I _{OH} = -2.0 mA	5
		5.5V	V _{CC} -0.4		4.8	V	I _{OH} = -2.0 mA	5
		4.5V	V _{CC} -0.4			V	Low Noise @ I _{OH} = -0.5 mA	
		5.5V	V _{CC} -0.4			V	Low Noise @ I _{OH} = -0.5 mA	
V _{OL1}	Output Low Voltage	4.5V		0.4	0.1	V	I _{OL} = +4.0 mA	5
		5.5V		0.4	0.1	V	I _{OL} = +4.0 mA	5
		4.5V		0.4	0.1	V	Low Noise @ I _{OL} = 1.0 mA	
		5.5V		0.4	0.1	V	Low Noise @ I _{OL} = 1.0 mA	
V _{OL2}	Output Low Voltage	4.5V		1.0	0.3	V	I _{OL} = +12 mA,	5
		5.5V		1.0	0.3	V	I _{OL} = +12 mA,	5
V _{OFFSET}	Comparator Input Offset Voltage	4.5V		25.0	10.0	mV		
		5.5V		25.0	10.0	mV		
V _{LV}	V _{CC} Low Voltage Auto Reset		1.8	3.8	2.8	V	@ 6 MHz Max. Int. CLK Freq.	3
I _{IL}	Input Leakage (Input Bias Current of Comparator)	4.5V		-1.0	1.0	µA	V _{IN} = 0V, V _{CC}	
		5.5V		-1.0	1.0	µA	V _{IN} = 0V, V _{CC}	
I _{OL}	Output Leakage	4.5V		-1.0	1.0	µA	V _{IN} = 0V, V _{CC}	
		5.5V		-1.0	1.0	µA	V _{IN} = 0V, V _{CC}	
V _{ICR}	Comparator Input Common Mode Voltage Range		0	V _{CC} -1.5		V		

DC ELECTRICAL CHARACTERISTICS (Continued)

Sym	Parameter	V _{CC} [4]	T _A = -40°C to +105°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max				
I _{CC}	Supply Current	4.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 2 MHz	5,7
		5.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 2 MHz	5,7
		4.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 8 MHz	5,7
		5.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 8 MHz	5,7
		4.5V		20.0	12.0	mA	All Output and I/O Pins Floating @ 12 MHz	5,7
		5.5V		20.0	12.0	mA	All Output and I/O Pins Floating @ 12 MHz	5,7
I _{CC1}	Standby Current	4.5V		5.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	5,7
		5.5V		5.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	5,7
		4.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 8 MHz	5,7
		5.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 8 MHz	5,7
		4.5V		7.0	4.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 12 MHz	5,7
		5.5V		7.0	4.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 12 MHz	5,7
I _{CC}	Supply Current (Low Noise Mode)	4.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 1 MHz	7
		5.5V		11.0	6.8	mA	All Output and I/O Pins Floating @ 1 MHz	7
		4.5V		13.0	7.5	mA	All Output and I/O Pins Floating @ 2 MHz	7
		5.5V		13.0	7.5	mA	All Output and I/O Pins Floating @ 2 MHz	7
		4.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 4 MHz	7
		5.5V		15.0	8.2	mA	All Output and I/O Pins Floating @ 4 MHz	7

Sym	Parameter	V _{CC} [4]	T _A = -40°C to +105°C		Typical @ 25°C	Units	Conditions	Notes
			Min	Max				
I _{CC1}	Standby Current (Low Noise Mode)	4.5V		4.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 1 MHz	7
		5.5V		4.0	2.5	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 1 MHz	7
		4.5V		4.5	2.8	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	7
		5.5V		4.5	2.8	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 2 MHz	7
		4.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 4 MHz	7
		5.5V		5.0	3.0	mA	HALT Mode V _{IN} = 0V, V _{CC} @ 4 MHz	7
I _{CC2}	Standby Current	4.5V		20	1.0	μA	STOP Mode V _{IN} = 0V, V _{CC} WDT is not Running	7,8
		5.5V		20	1.0	μA	STOP Mode V _{IN} = 0V, V _{CC} WDT is not Running	7,8
I _{ALL}	Auto Latch Low Current	4.5V		40	16	μA	0V < V _{IN} < V _{CC}	
		5.5V		40	16	μA	0V < V _{IN} < V _{CC}	
I _{ALH}	Auto Latch High Current	4.5V		-20.0	-8.0	μA	0V < V _{IN} < V _{CC}	
		5.5V		-20.0	-8.0	μA	0V < V _{IN} < V _{CC}	

Notes:

1. Port 2 and Port 0 only
2. V_{SS} = 0V = GND
3. The device operates down to V_{LV} of the specified frequency for V_{LV}. The minimum operational V_{CC} is determined on the value of the voltage V_{LV} at the ambient temperature. The V_{LV} increases as the temperature decreases.
4. V_{CC} = 4.5V to 5.5V, typical values measured at V_{CC} = 5.0V
5. Standard Mode (not Low EMI Mode)
6. Z86E08 only
7. All outputs unloaded and all inputs are at V_{CC} or V_{SS} level.
8. If analog comparator is selected, then the comparator inputs must be at V_{CC} level.

APPENDIX F
USER MANUAL

USER MANUAL

for

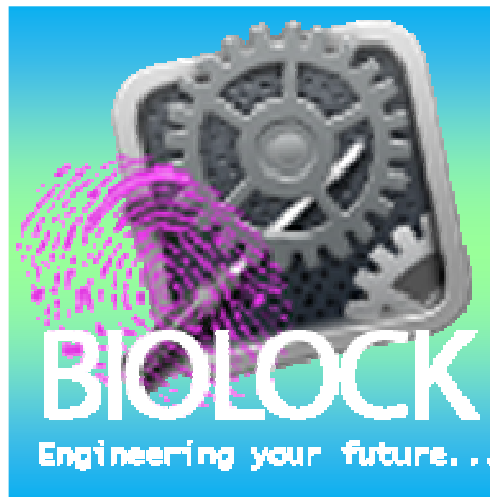
**Automated Finger Print
Activated Door Lock**

Prepared by

Eliseo G. Noble Jr.
Jean Eric V. Agena
Jon Remon D. Loon
Judy Ann U. Rodriguez
Karl Lester A. Co

COE 461D / C1

March 17, 2008



Welcome to Automated Finger Print Activated Door Lock (AFPADL) Application Software. This software is under a BIOLOCK trademark. This program is made for the purpose of enabling the automatic locking and unlocking of a door. The software also includes additional functions like Administration Control with Administrator log-in authentication.

This application software, AFPADL, is exclusively intended for the use of demonstrating the functionalities of the team's design. Any form of reproduction, transfer, distribution or storage of part or all of the contents in this document without the prior written permission from the team or the authors is prohibited.

General Information

- About your software

This Software Design Document provides a complete description of the design of the application software for Automated Finger Print Activated Door Lock (AFPADL), developed for the BIOLOCK team's design. The dominant design methodology is an object-oriented design using a Visual interface to a database management system.

One part of the system is made for activating the door enabling it to lock and unlock. This part requires a finger reading for the authentication of the access to the room. Another part of the system is available for the administrators' control over the design. This function is capable of adding, deleting and updating the records and schedules of the persons who can access the room.

The user will do most normal maintenance of the persistent data in the database using database utilities. These include adding and deleting of professors and their finger prints, edit or update schedules and monitoring the daily access to the room.

The user accesses this system through forms. These forms interact with several code modules to provide the bulk of the services. In turn these code modules interact with the underlying database.

This system is designed to run only in 1 computer terminal. It is a stand-alone application which does not require any internet connection or networking capabilities but needs a serial port or a USB to serial port plug.

- Overview of the functions of the software

This software provides many functions that suffice AFPADL's requirements. It is made practical and easy to use because the database is very user friendly and its appearance is designed to hold multiple records. Some of the functions of the system are

- Activating the locking and unlocking of the door.
- Adding, Deleting and Editing of Administrators.
- Adding, Deleting and Editing of Professors.
- Registering finger prints for professors.
- Scheduling the access to rooms of the professors.
- Adding and Deleting of Rooms, Subject and Department.
- Monitoring the daily access to the room.
- Force Controlling of the door.
- Log-in authentication for Administration Controls.

- System Requirements

This system includes modules that can only run in certain platforms. Minimum and recommended requirements are mentioned below.

Hardware Requirements:

- Pentium II or higher (Pentium III recommended)
- 128 MB RAM (256 MB RAM recommended)
- 350 MB free hard disk space or higher
- Serial Port or USB to RS-232 Plug

Software Requirements:

- Microsoft Windows XP, 2000 SP4 or later
- Microsoft .NET Framework 2.0
- Microsoft SQL Server Native Client
- Microsoft SQL Server

- **Access codes**

The log-in with password code helps to protect the confidentiality of the administrator controls against unauthorized use. The default codes would be given together with this document, it is up to the user to create another account and password then delete the default user administrator to prevent illicit use of this software together with its contents.

Getting Started

- **Installing the software**

To install the software, be sure that your computer is turned on and its CD ROM drive is present within its components. Just follow these steps to properly install the program.

1. Insert software CD, AFPADL, to the CD ROM drive.
2. A pop up window would automatically appear.
3. Once a pop up prompts that contains the "Setup.exe". Double click the icon and follow the steps in installing the software. Install a MS SQL Server and attach the database included in the Installation CD.
4. Close the Pop up window.
5. Locate the .EXE file and double click.
6. The software is now ready to use.

- **Starting the software**

To start the software, of course, your computer must be turned on. These steps should be followed to start the software:

1. Turn on the Voltage Regulator.
2. Plug in the cord of the Monitor, the Central Processing Unit (CPU).
3. Turn on your CPU by pushing the "ON" button.
4. Wait for your computer to start-up.
5. After the start-up, if your computer is password-protected, and then write in your password and login.
6. Find the Executable Icon of the software.
7. Double-click the .exe file to start using the software.
8. Make sure that the finger print scanner and the circuit is attached to the computer and the Database Server is running.

- **Main Menu**

As soon as the AFPADL starts, the main menu window will appear. This window will be your gateway to the different features of the software. If a menu item is clicked it automatically brings you to other window forms. The main window is composed of 2 menu items.



There are two features available in the Main Window:

1. **Administrator Control** – displays the administrator's control features and functions after a successful Log-in.
2. **Activate System** – displays the form for the identification of the persons who is accessing the room.

- **Logging in**

To log-in, you must do the following steps:

1. Type the username in the space provided.
2. Type the password in the password space provided. (NOTE: the password is case sensitive, so if you used capital letters be sure to type them in capital letters.)
3. Click the ok button to log-in to the system.

4. Remember that an individual can not utilize the feature of an administrator if he/she is not logged in.



- Logging out

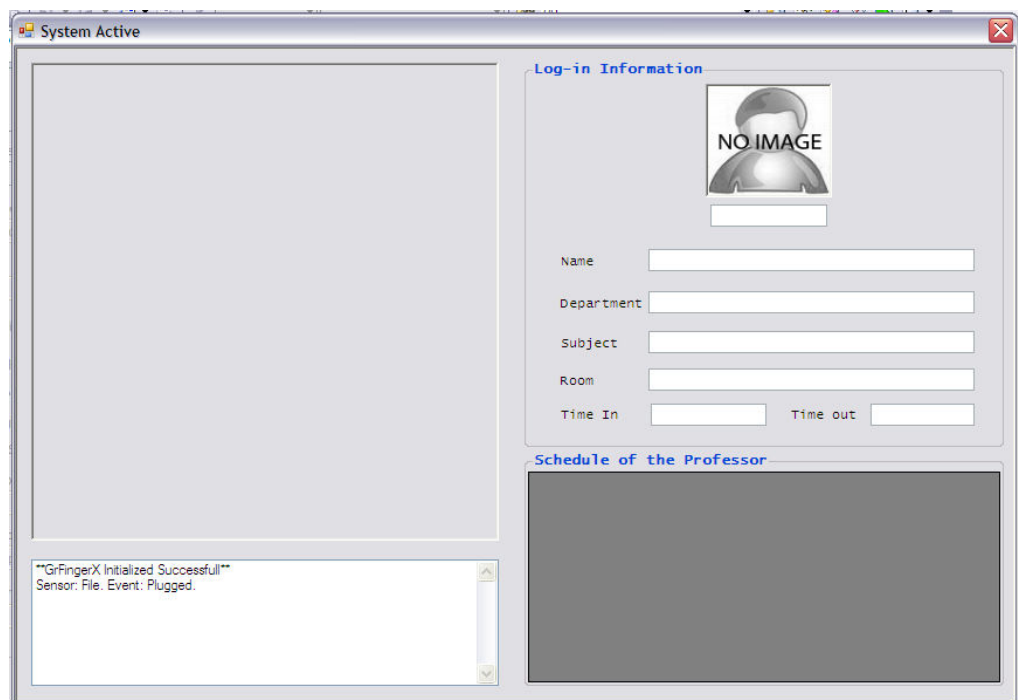
To log-out just press the Logout button under the Administrator Control menu. It is necessary to log-out your account before exiting the program.

- Administrator Control Window



As previously mentioned, as soon as the Administrator Control button is clicked from the main window, log-in button will appear. After it was clicked, the Log-in form will appear. An administrator has to log-in first before he/she can be able to use the functions available for the administrator. After a successful Log-in, the administrator can now be able to add an administrator or change his own password. He/She can add, edit or delete a professor, give them a room access schedule and let them record the professors' finger prints. Another feature available for the administrator is to add and delete a room, department or a subject. An administrator can also view the daily access on a room. The administrator can also force an opening (unlocking) or closing (locking) of a room door.

- Activate System Window



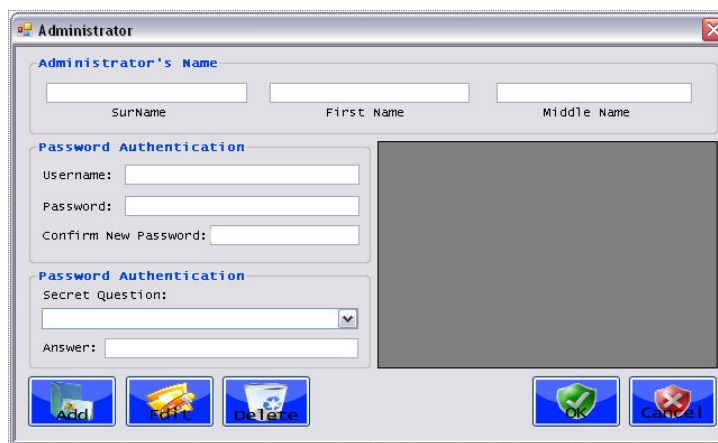
This window will appear after “Activate System” button is clicked. The Activate System window is used to initialize the biometric device specifically the finger print scanner. After a professor has been given a schedule and has already recorded his finger prints, he/she can now access the room depending on the schedule given to him/her. His

profile will be showed along with his print. A valid access to the room can either lock or unlock a door depending on time he/she is allowed to enter his/her finger print.

Administrator Control Menu Item

- Administrator

is composed of functional tools to add, delete an administrator.



The image shows a Windows-style dialog box titled "Administrator". It contains several input fields and buttons. At the top, under "Administrator's Name", there are three text boxes labeled "SurName", "First Name", and "Middle Name". Below this is a "Password Authentication" section with three text boxes: "Username:", "Password:", and "Confirm New Password:". Underneath is another "Password Authentication" section with a "Secret Question:" dropdown menu and an "Answer:" text box. On the right side of the dialog is a large, empty rectangular area. At the bottom, there are five buttons: "Add" (with a plus icon), "Edit" (with a pencil icon), "Delete" (with a trash can icon), "Ok" (with a green checkmark icon), and "Cancel" (with a red X icon).

Add button – assuming that all required fields for an administrator are filled in after add button is clicked the administrator is added to the administrator list.

Edit button – if an item is clicked in the list edit button will be enabled. When clicked, password can be changed and by clicking the Save button, the administrator's changes done will be saved.

Save button – saves the changes done by the administrator.

Delete button – if an item is clicked in the list delete button will be enabled. When clicked, the item selected in the list will be deleted.

Ok button – if all is done in the form click this to exit the form.

Cancel button – exits the form.

- **Professor**

is composed of text boxes, picture box, combo boxes and browse button to fill – up the fields in a professor profile. This also contains a button for adding and deleting a schedule for the professor as well as a button for registering the professor’s finger print.

The screenshot shows a software window titled "Professor". Inside, there's a section for "Professor's Name" with three text boxes labeled "SurName", "First Name", and "Middle Name". Below this is a "Profile" section containing a placeholder for a profile picture (labeled "NO IMAGE") with a "Browse..." button underneath it. Further down are text boxes for "Identification No:" and "Department:". At the bottom of the window are four buttons: "Delete" (with a trash icon), "Cancel" (with a red X icon), "Add Schedule" (with a calendar icon), and "Register Print" (with a fingerprint icon).

Add button – assuming that all required fields for a professor are filled in after add button is clicked the professor is added to the professors list.

Delete button – if an item is clicked in the list delete button will be enabled. When clicked, the item selected in the list will be deleted.

Add Schedule button – this feature will add a schedule for the professor to access the room.

Register Print button – saves finger print of the professor to access the room in a given schedule.

Ok button – if all is done in the form click this to exit the form.

Cancel button – exits the form.

- **Items**

is composed of functional buttons to add and delete rooms, department and subject.

The 'Items' window has a title bar with a close button. It contains three tabs: 'Rooms', 'Subject', and 'Department'. The 'Rooms' tab is selected. Inside the 'Rooms' tab, there are three input fields: 'Room:', 'Lock Address:', and 'Room Sensor:'. Below these fields is a large, empty rectangular area, likely for a list of items. At the bottom of the window, there are four buttons: 'Add' (with a folder icon), 'Delete' (with a trash can icon), 'Ok' (with a green checkmark icon), and 'Cancel' (with a red X icon).

Add button – upon loading, the Add button is already activated. Once clicked, assuming that fields for adding rooms, subject or department are filled in the information entered will then be added to the database.

Delete button – this button removes the selected item from the list from the database.

Ok button – if all is done in the form click this to exit the form.

Cancel button – exits the form.

- **Room Access**

is composed of combo box, date time picker and buttons to search for records of access to a room in a specific date.

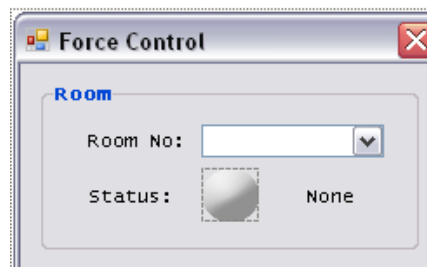
The 'Room Access' window has a title bar with a close button. Below the title bar is a section labeled 'Information'. Inside this section, there are two dropdown menus: 'Room No:' and 'Date:'. The 'Date:' dropdown is currently set to 'March 11, 2008'. To the right of these dropdowns are two buttons: 'Search' and 'Back'. Below the 'Information' section is a large, empty rectangular area, likely for displaying search results.

Search button – assuming that a room is selected from the given choices and a date for searching is specified; once search button is clicked it will search for records of access to the room for that day.

Back button – returns to the administrator control menu once clicked.

- **Force Control**

is composed of a combo box and a status button to indicate the status of the door.



Status button – upon loading, the list of available rooms are shown in the combo box. After selecting a specific room the status button will indicate whether the room is locked or unlocked. If the status is green it means that the door is unlocked. If the status is red it suggest otherwise. When clicked it complements the status of the door which does not require any finger print scanning.

Operating Procedures

- **Administrator**

- **Add**

1. To add a new administrator, click add button.
2. Administrator fields should be filled in first to continue adding.
3. In any case that required fields are left empty a message prompt will appear to notify the user to fill in empty fields.
4. To cancel adding of administrator, use the cancel button.
5. To add an administrator again, repeat step 1.

- **Edit**

1. Editing or changing of password is available for the administrator.

2. To edit the administrator's password select the logged in administrator's name from the list then click edit button.
 3. Take note that only the logged-in administrator can change password.
 4. Click Save button to save changes.
- Delete
 1. To delete an administrator select the name of the administrator to delete from the list.
 2. Click delete button to remove his name from the administrators database.
 - Professor
 - Add
 1. To add a new professor, click add button.
 2. Professor's profile fields should be filled in first to continue adding.
 3. In any case that required fields are left empty a message prompt will appear to notify the user to fill in empty fields.
 4. To cancel adding of professor, use the cancel button.
 5. To add a professor again, repeat step 1.
 - Delete
 1. To delete a professor, select the name of the professor to delete from the list.
 2. Click delete button to remove his name from the professors database.
 - Add Schedule
 1. To add a schedule of access to the room for the professor, select a professor from the list.
 2. Click "add schedule" button to add a schedule for the professor.
 3. A new window will appear.
 4. Fill in required fields then click Add button to save the schedule, cancel to return to the professor window or delete to delete a specific schedule from the list.
 - Register Print
 1. To register the professor's finger print to access the room to his/her given schedule, click Register print button.
 2. A new window will appear.
 3. Place the professor's finger print on the finger print scanner.
 4. Click add button to register his/her finger print.
 5. Do this several times depending on the required finger print adding for a professor. To do this just repeat steps 3 and 4.
 6. Click close button at the upper right corner of the form to return to the professor window.

- **Items**
 - **Add**
 1. To add a room, department or subject, select a choice from the tab control or from the given choices (room, subject or department).
 2. Fill in required fields then click Add button to add that item to the database.
 3. Click OK or Cancel button to return to the Administrator Control menu.
 - **Delete**
 1. To delete a room, department or a subject, select it from the list depending on the chosen tab control.
 2. Click delete button to remove that item from the list and from the database.
 3. Click OK or Cancel button to return to the Administrator Control menu.
- **Room Access**
 - **Search**
 1. To search for a room access record, choose a room from the drop down list and select a specific date.
 2. Click Search button to search for the room access record.
 3. If no record exists for that date and room a prompt will appear to notify that no match is found.
 4. To search again, repeat step 1.
 5. To go back to the administrator control menu click Back button.
- **Force Control**
 - **Force a door to lock or unlock**
 1. To lock or unlock a door without finger print reading, click the force control button from the administrator control menu.
 2. Choose a room to lock or unlock.
 3. Click the status button to lock or unlock the door.
 4. Remember not to leave the door open for a long time for this might cause a malfunction to the system.
 5. To go back to the administrator control menu click the close button from the upper right corner of the window.
- **Activate System**
 - **To activate the system**
 1. Click Activate system from the Main Menu.
 2. The professor can now enter a room by placing his/her finger print from the finger print scanner.
 3. To close the activate system window click the close button from the upper right corner of the window.